NASA-CR-195746

College of Science and Technology

170 p

usm

(NASA-CR-195746) TECHNOLOGY DEVELOPMENT SUPPORT FOR THE TECHNOLOGY TRANSFER PROGRAM AT THE INDIANAPOLIS CENTER FOR ADVANCED RESEARCH. VOLUME 2: REVIEW OF PREVIOUS YEAR'S WORK Final Report, Jul. 1984 - Jun. 1985 (University of Southern Mississippi) 170 p

N94-71793

Unclas

Z9/61 0002858

SCIENCE AND TECHNOLOGY

TECHNOLOGY DEVELOPMENT SUPPORT FOR THE TECHNOLOGY TRANSFER PROGRAM AT THE INDIANAPOLIS CENTER FOR ADVANCED RESEARCH

VOLUME II

TECHNOLOGY DEVELOPMENT SUPPORT FOR THE TECHNOLOGY TRANSFER PROGRAM AT THE INDIANAPOLIS CENTER FOR ADVANCED RESEARCH

VOLUME II

REVIEW OF PREVIOUS YEAR'S WORK

G. David Huffman
W. Leigh

July, 1985

College of Science and Technology
The University of Southern Mississippi
Southern Station Box 5165
Hattiesburg, Mississippi 39406-5165

EXECUTIVE SUMMARY

The University of Southern Mississippi's Department of Computer Science in conjunction with the Indianapolis Center for Advanced Research has been conducting a comprehensive research program with the objective of developing a series of software systems for use in technology transfer activities. The all encompassing set of software tools includes both commercially available and custom software and covers the complete spectrum of technology transfer tasks, i.e., search formulation, analysis of abstracts for relevancy, report outlining and report preparation.

The software systems were orginally developed for use in a super-mini-computer environment, i.e., DEC VAX 11/780, but are now being reconfigured for use on micro-computers having disk capability. The final report covers all activities of the project in considerable depth and consists of a collection of publications generated during the current contract year--July, 1984 to June, 1985. In summary, substantial progress has been made and the research team intends to install demonstration versions of the software at selected NASA Industrial Application Centers and State Technology Assistance Centers during the coming year.

TABLE OF CONTENTS

OVERVIEW

The approach taken in this final report differs somewhat from that used in the past. Rather than distilling and presenting an overview of activities, a collection of all papers prepared, presented and/or published has been included. A strong research group has now been assembled and the quality and quantity of the research is impressive. Eight individuals are now involved in various aspects of the software research and development. Only three of these received any support from the sponsoring organization, i.e., Dr. Huffman-25%, Dr. Leigh-33% and Mr. Vital-50%. The remaining individuals are supported by their home institution, i.e., the University of Southern Mississippi and Northern Kentucky University. The net result of the increased level effort is a strong porgram dealing with many aspects of automating the technology transfer process.

The first document, "Computer Applications in Information Retrieval and Writing for Technology Transfer," by Leigh and Berry, is to appear as a chapter in COMPUTER APPLICATIONS FOR THIRD-WORLD COUNTRIES, to be published by MacMillan in 1985. This article describes the process of technology transfer as it is carried out using bibliographic database searching.

"Applying Natural Language Techniques to Improve Management Information Systems," by Paz and Leigh, was written to describe how human-communication interaction methods developed in the context of natural language interface systems might be applied to database retrieval and information systems in general and in formats not considered to be natural language. The contents of this report are applicable to bibliographic database searching systems as well. This article was submitted to the JOURNAL OF SYSTEMS MANAGEMENT.

"Software Support Systems for Use in Technology Transfer Activities," by David Huffman, reflects our efforts to identify commercial personal computer software which might be applied to the database searcher's tasks. This article has been published by the JOURNAL OF TECHNOLOGY TRANSFER, Volume 9, 1985, pages 29-42.

"A PC-Based Gateway for Information Retrieval," by Leigh and Souder, reports our development of a prototype program which implemented a menu-interface common command language for database searching. This paper appeared in the PROCEEDINGS of the Second Ergonomics and Human Factors Conference, June, 1985.

"A Mechanism for Matching Multiple Patterns in a Text String," by Violet Chen, is a M.S. Project report submitted as a requirement of receiving the degree. Ms. Chen extended some basic string-matching algorithms to function with multiple, optional pattern occurrences. This work has direct application to problems found in translating the formats of bibliographic citations.

"Information Retrieval and Special Attention Paid to Hypertext," is a M.S. Project report submitted by Steve Howard. Mr. Howard built a prototype hypertext-type citation classifying system. From this work, we learned that such an approach is feasible and useful for technology transfer applications.

"Development of a Technology Transfer Workstation," by Leigh, Huffman, Paz, and Vital, is an overview of our whole project to this point. This paper has been submitted to the Hawaii International Systems Science Conference, to be held in January of 1986.

On a relevant but unrelated topic, David Huffman presented a paper at the International Technology Transfer Symposium entitled "Economic Development and Technology Transfer: Mississippi's Approach."

The names of several people appear as authors of the above papers and articles. William Leigh and G. David Huffman, of course, are the principal investigators for the project. Noemi Paz (University of Southern Mississippi), Michael Berry (Northern Kentucky University), and Ray Souder (Northern Kentucky University) are colleagues of the principal investigators. They have participated in this work as part of their own research programs. Violet Chen, Steve Howard, and Dennis Vital are graduate students at the University of Southern Mississippi who have assisted with the project in meeting their graduate school degree requirements.

The M.S. thesis of Dennis Vital has not been finished in time to appear in this annual report. This document will contain complete documentation and source code for the SORT-AID system, which has been a major software development endeavor of this project. This thesis will contain comparative timings for the SORT-AID citation classification aid on the VAX computer and on the IMB-PC.

Computer Applications in Information Retrieval and Writing for Technology Transfer

William Leigh
University of Southern Mississippi
Michael Berry
Northern Kentucky University

To appear as a chapter in
COMPUTER APPLICATIONS FOR THIRD-WORLD COUNTRIES
to be published by MacMillan in 1985

Computer-Aided Research and Writing

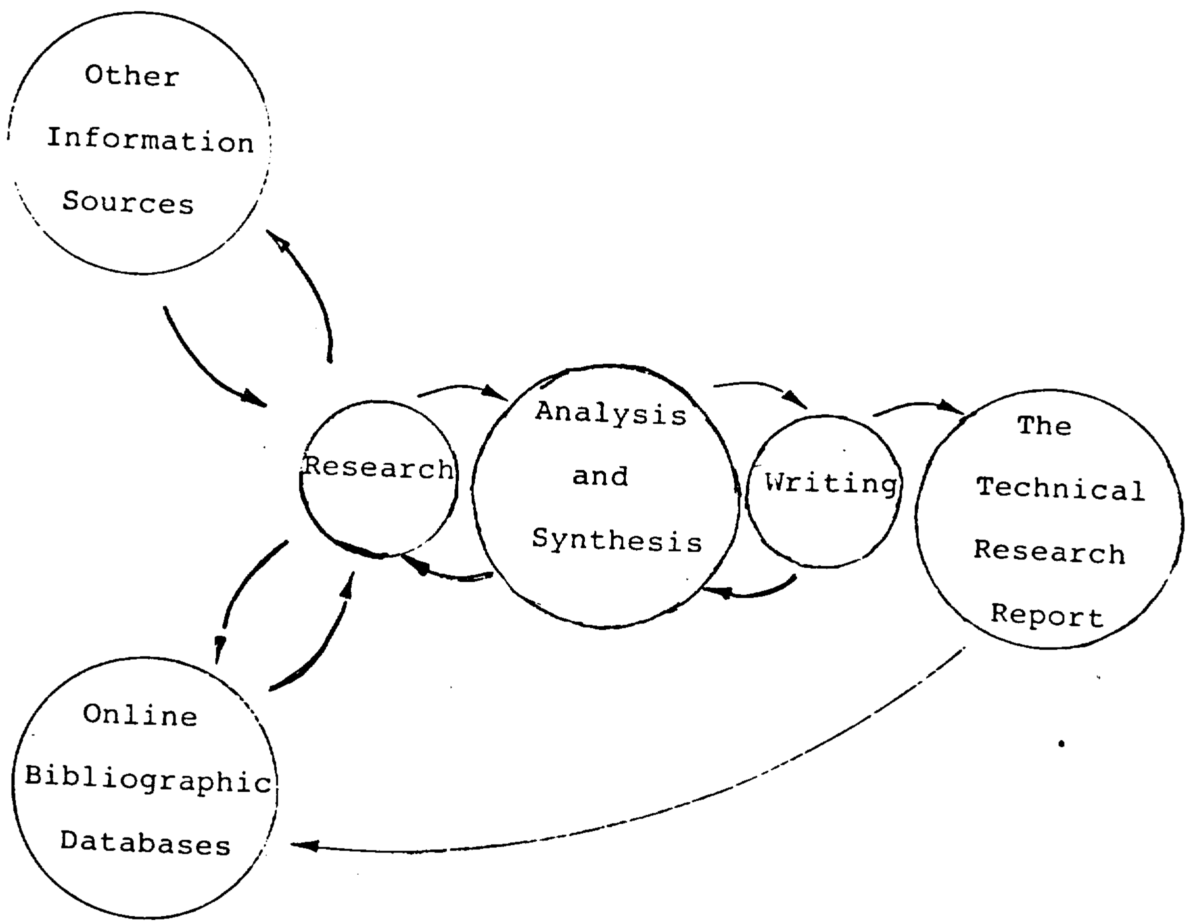
The computer, and especially the personal computer, has potential as an aid to the type of knowledge work known as research and writing. This article describes this capability as it exists now and suggests where this capibility will be extended in the future. The scope embraced includes the processes of composing (information collecting, organizing, and synthesizing) and producing (text keyboarding, editing, and formatting) research reports, especially those reports which involve bibliographic searches as in technology transfer. Figure 1 is a block diagram of the flow of the processes of composing and producing research articles and reports. The computer can be a tool to accelerate and augment these processes, or it can be a full-fledged assistant (1).

Developing nations have acute needs to transfer technology. An effective tool for this task, which has become possible with computers, is accessing technical bibliographic databases. This chapter describes how computers may be used to enhance that process of technology transfer which is accomplished by information retrieved and technical report writing.

Information Retrieval

Since the 1960's, when the enabling computer technology became available, many bibliographic citation and full-text databases covering almost any area of knowledge have been

Figure 1: Flow of Information and Tasks in the Composition and Production of a Technical Research Report.



built and made available for access with computer terminal equipment via telephone lines. Figure 2 is a diagram of this arrangement. Lancaster (2) supplies a good overview of the history and capabilities of information retrieval systems. Salton (3) offers a survey of technical developments in the field.

Bibliographic databases contain citation entries composed of titles, authors, publication information, and abstracts of articles, books, and reports. Full-text databases contain the complete texts of the documents in addition. There is one entry in these databases for each document. Usually the databases specialize in one area of knowledge. Glossbrenner (4) is a directory to the most widely available databases, what they contain, and how they are accessed. Table 1 contains a list of some major databases and a description of their contents. Wente (5) describes one bibliographic database system in detail.

Entries are accessed with keywords. Keywords are supplied for the documents by the author, by a human indexer, or with automatic methods. Figure 3 contains a citation including keywords which might be from a computer science bibliographic database.

Queries are formulated by the researcher to access databases. There queries must be expressed in the special language supported by the specific database. Efforts are underway to standardize these languages across databases.

The International Organization for Standardization (ISO) is considering one such standard language as described in the "Draft Specification for Commands for Interactive Search Systems" (6). Figure 4 contains several queries expressed in this draft standard query language. Another solution to the problem of multiple querying languages which does not involve changing the database systems themselves is a "gateway" approach. Marcus (7) has built such a system which uses a computer to convert queries specified in a standard language to the specific languages of the individual database systems. Figure 5 shows the relationship of the components in the "gateway" approach to online information retrieval.

The major skill required in formulating queries is picking the keywords to specify for the search. Not only must the keywords chosen match the requirements and interests of the searcher, but they must also be consistent with the indexing vocabulary of the database. An option which is becoming more popular as computer processing power becomes cheaper is the full-text search for keywords. Rather than confining the search to the keywords formally assigned to the documents, the searcher instructs the system to search the abstracts, or even the documents' full texts, for the presence of multiple words which occur in a specified proximity.

Figure 2: Online Information Retrieval

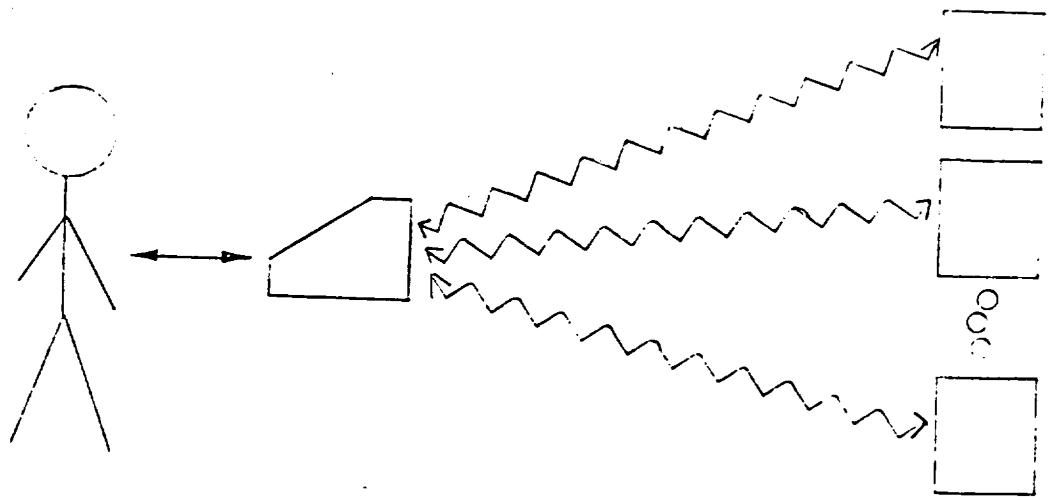


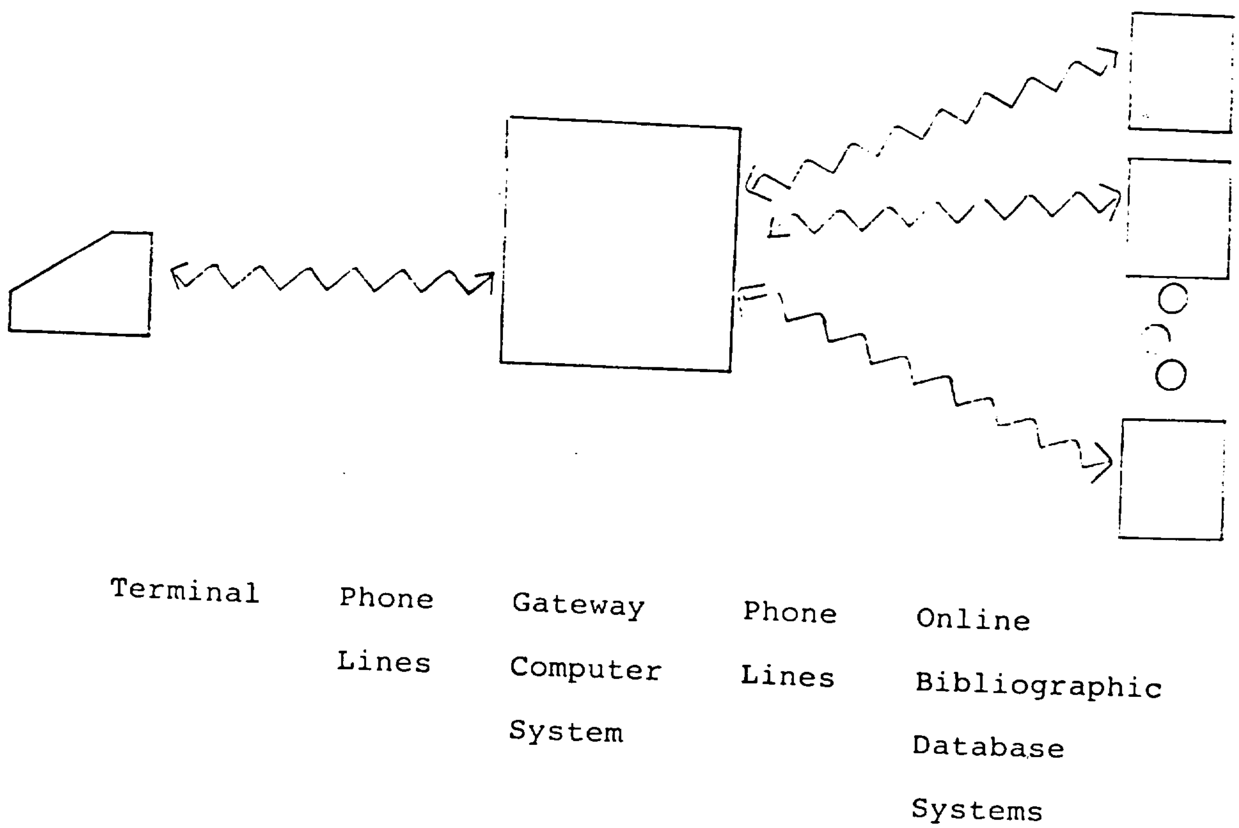
Table 1: Some Major Databases and Their Subjects

ABI/INFORM	Business and Economics
AGRICOLA	Agriculture and Related Subjects
COMPENDEX	Worldwide Coverage of Engineering
ERIC	Educational Materials
INSPEC	Physics, Electrotecnology, Computers
MEDLINE	Medicine, Dentistry, Nursing

Figure 3: Citation from a Typical Bibliographic Database

Title	- A Data Structure for Word Processing
Author	- Jones, John M.
Date	- August, 1982
Source	- Journal of Computer Science
Pages	- 354-368
Keywords	- DATA STRUCTURES, WORD PROCESSING, OFFICE AUTOMATION, TEXT PROCESSING, TEXT EDITING
Abstract	- A data structure is presented which has desirable properties in situations requiring automatic justification under conditions of insert and over-write. This situation occurs in screen-oriented word-processing systems. Examples of use and test results are reported.

Figure 5: Diagram of a "Gateway" Architecture.



Modern information retrieval systems offer facilities for rapid and exhaustive generation of bibliographies and

Figure 4: Example Queries.

```
FIND WORD PROCESSING AND DATA STRUCTURES
FIND (WORD PROCESSING OR TEXT PROCESSING) AND TEXT EDITING
FIND (DATA STRUCTURES OR ALGORITHMS) AND WORD PROCESSING
FIND TEXT EDITING AND DATE 1977
```

location of source materials for research. This is a use of computers for accelerating knowledge work. Developments in this area are beginning to make this capability available to knowledge workers who cannot invest in the training and experience now required to access multiple databases effectively. Such improvements in the man-machine interfact of information retrieval systems are taking the form of standard, more "user-friendly" languages and "gateway" computer system architectures.

Review and Analysis of Citations

Citations once retrieved from a database may be delivered to the searcher on paper or "down-loaded" in computer-readable form to his "gateway" or personal computer. The task now is to determine which citations are relevant and which citations are promising enough to justify ordering the document, and then to classify the relevant citations into sub-areas of the research topic. The computer can accelerate this task. Some aspects of this task may even be delegated to the computer.

Figure 6 shows an example of a terminal display screen which might be presented by a typical citation review program. At the bottom of the screen is a "menu" of possible operations to be selected by the reviewer. According to that menu, pressing the RETURN key presents the next citation in the file. Selecting the "o" option allows the operator to assign categories to the citation. These categories might be topic-specific assigned at the

Figure 6: Example of REVIEW Terminal Display.

```
Title    - A Data Structure for Word Processing
Author   - Jones, John M.
Date     - August, 1982
Source   - Journal of Computer Science
Pages    - 354-368
Keywords - DATA STRUCTURES, WORD PROCESSING,
           OFFICE AUTOMATION, TEXT PROCESSING,
           TEXT EDITING
Abstract - A data structure is presented which has desirable
           properties in situations requiring automatic
           justification under conditions of insert and
           over-write. This situation occurs in screen-
           oriented word-processing systems. Examples of
           use and test results are reported.
------------------------------------------------------------
USER-ASSIGNED CATEGORIES:  INTRODUCTORY, ORDER

------------------------------------------------------------

MENU:  RETURN   - next citation
       0        - assign categories
       1        - search on literal
------------------------------------------------------------
```

discretion of the reviewer or they might be of other significance, such as "ORDER" for a document to be ordered in its full-text version. The "1" option allows the reviewer to search the citation file for the presence of words or phrases.

The possibility of an automatic method of determining the relevance of documents or abstracts has been an object of research in information science for some time (8). One approach to this uses the lexical association methods which have their orgin in the work of Luhn (9). Luhn observed that the significant terms in a document were neither the most frequent nor the least frequent terms in the document. Other researchers have developed this observation and its derivatives into methods for automatic abstracting and automatic keyword generation (10).

As the abstracts which are the result of a search are already homogeneous in content, the most frequently appearing words in the collection (after discarding a "stop list" of common words which have little meaing such as "a", "an", "the", and so forth) should indicate the overall nature and purpose of the search. So abstracts which contain these words can be considered consistent with the purpose of the search, and therefore relevant. A ranking of the members of the set of citations resulting from a search may be accomplished by considering as more relevant the citations whose abstracts contain more of these frequent words. This method can be more effective if the frequent words to be used in the ranking are first displayed on the terminal display for the operator to review and discard if, by chance, they are not relevant as intended.

After the citations are ranked, they can be reviewed in order with the REVIEW facility described above. It is desirable that they be reviewed in order of decreasing relevance score. This allows the researcher, when he is satisfied that enough citations to terminate the reviewing session with some assurance that the remaining citations are of lesser relevance than the ones reviewed directly.

Automatic relevance ranking is an example of the delegation of some of the substance of the research process to the computer. Such a computer-aided process as this can be effective in reducing the work required to manually review many abstracts. This type of pre-ranking is only feasible with the aid of a computer.

Organization of Information and Ideas

It is in the organization and synthesis of information to promote the generation of ideas that the computer may eventually make its main contributions. Bush (11) anticipated this use of a computer technology in 1945, altghough such technology did not exist then. Bush's MEMEX system, as it was proposed, allowed the structuring of documents into an arbitrarily deep outline structure and the reuse of document segments by linking them into new documents with "trails," or connections between similar subjects in different documents.

The MEMEX ideas have been re-proposed repeatedly and have achieved many implementations. A recent implementation is Price's THUMB system (12). In the THUMB system, experts prepare multi-threaded indices for documents so that readers can peruse the document at different levels of detail and investigate diferent aspects directly.

The THUMB system is an example of how a computer-based system can augment the powers of a researcher in the substance of his work. Expertly prepared THUMB indices can present a document to a new reader in a manner which directly meets his needs.

"Knowledge graphs," which is a modern term for the types of structures used by THUMB, can become the basis for a "knowledge support environment" (13). In the context of the analysis and organization of bibliographic citations, the automatic creation of a knowledge graph for a retrieved citation set would facilitate the understanding of a field of knowledge and the interactions between and among the documents. It is not likely that this can be accomplished effectively with lexical association or syntactical methods. The realization of this ambition will wait for the development of adequate natural language semantic understanding methods in artificial intelligence. However, these structures might be developed manually for textbooks or for seminal technical articles, or a researcher could build such a structure to support his own personal endeavors.

One application for knowledge graphs which has not been investigated is as a vehicle for the presentation of the keyword vocabulary of a bibliographic database. This could aid researchers in finding proper terms for searching and in becoming acquainted with the area of coverage of the database. Also, as the researcher followed the graph, he could designate keywords encountered as important to him or not. The system could formulate a search query from this trail.

The augmentation of human ability to organize informationand generate ideas is a great potential of the personal computer. this capability will need to be achieved by the availability of many different tools which can be selected for use and extended to the needs of individual researchers. This capability will only be achieved by a symbiosis between the personal computer and the research/writer.

Artificial Intelligence for Research and Writing

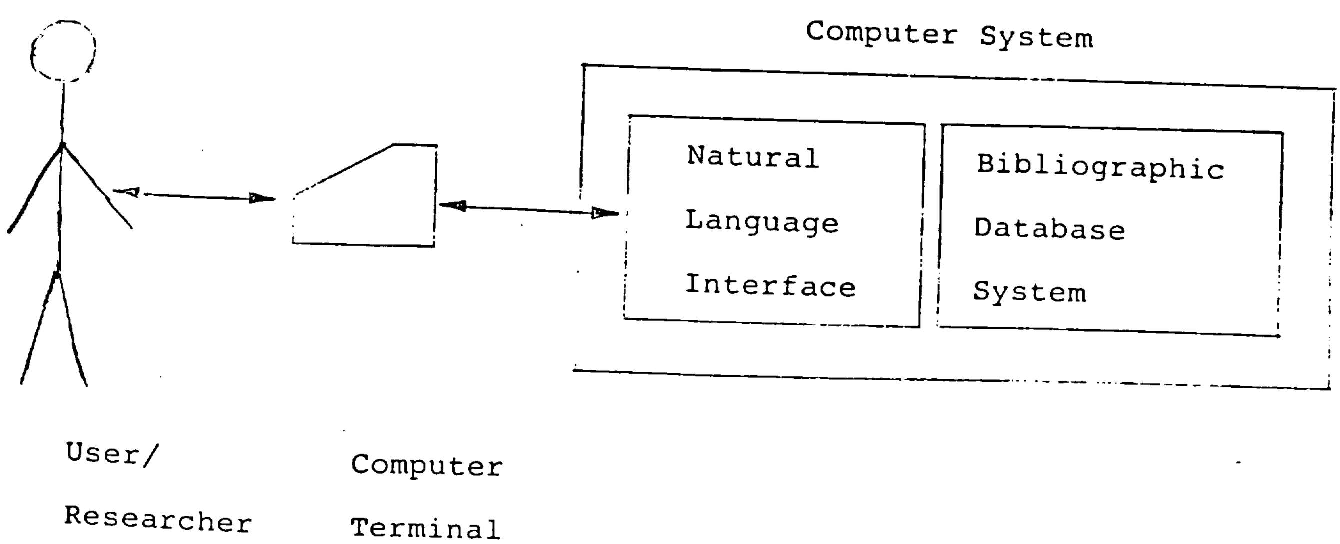
Machine translation between natural languages is an older ambition of artificial intelligence. Systems now exist which augment and accelerate a human translator's effectiveness. This is an example of the symbiotic type of man-machine cooperative system. The capability of semi- or fully automatic machine translation has obvious benefits for researchers and writers.

A special case of machine translation is the automatic answering of natural language questions directed against a database. A system which functions in this manner is a special type of "gateway," one which mediates between a natural language, such as English, and the formal query language of the database system. Figure 7 is a diagram of a possible relationship between the functional part of such a system.

A desirable attribute of natural language database question-answering systems is "cooperativeness" (14). This term denotes the behavior which answers the query "How many articles are about alternative energy sources?" not with "5" but with "There are only 5, but this database contains information only on agriculture." This type of behavior requires abilities beyond language translation. Self-knowledge and the facility to call it up when appropriate are needed.

Such a natural language question-answering system would be beneficial to researchers beyond conventional information retrieval. However, it will probably be some time before it is realized. With such systems of lesser skill, there would be considerable danger in attributing the system with greater knowledge and skill than it actually possesses. Artificial intelligence systems may seduce humans into suspending their own critical reasoning capacity. This is a greater problem because one of the greatest unsolved problems in artificial intelligence is to construct systems that know the limits of their own knowledge, that is, that know when they do not know.

Figure 7: An Organization for a System with a Natural Language Question-Answering Component.



An artificial intelligence technique which can be adapted to endow a system with self-knowledge is the "expert system." The expert system is close to wide-spread commercial application (15). An expert system has been built (16) to guide researchers in the selection of bibliographic databases for searching. Figure 8 shows some possible examples of "rules" which might be used in such a system. The computer-based expert system carries on a dialog with the user to determine the requirements of the search. These requirements are vetted against the rules to result in recommended databases.

The possibilities of artificial intelligence are exciting for researchers and writers. The future of such systems is not to replace but to augment human skills. Presently, such systems exist mainly in laboratories and in most cases can augment only the skills of the non-expert. Indeed, artificial intelligence systems may never truly augment the skills of real experts but only accelerate the accomplishment of some tasks. The great advantage to these systems may be in giving non-experts some semi-expert abilities through a working mode of symbiosis and mutual consultation.

Interactive Editing Systems

Screen-oriented text-editors are now commonly available with personal computers, and this type of system is nearly universally preferred; such a system allows the user to key and manipulate text on a computer screen and to see what the

Figure 8: Some Rules for a Computer-Based Expert system to Guide Database Selection.

IF SEARCH TOPIC IS AGRICULTURE THEN USE AGRICOLA

IF SEARCH TOPIC IS ELECTRONICS AND AGRICULTURE THEN USE AGRICOLA AND INSPEC

IF SEARCH TOPIC IS COMPUTERS THEN USE INSPEC OR COMPENDEX finished version will look like at all times.

Before the advent of the personal computer and its fast response and screen display, the standard for computer text editing was the command-oriented editor. Such systems were adapted to the constraints imposed by a connection to the computer over telephone lines which supported only slow data rates and by the then-standard terminal, the line-at-a-time typewriter terminal. Command-oriented, line-based editors refer to text by a computer-assigned line number. That line number must be referenced when modifying the text.

Screen-oriented editors allow the text to be modified directly on the screen. The particular text to be modified is pointed out with a curser character on the screen which is moved about with special cursor control keys or with a special auxiliary input device for this purpose. (One such special device for pointing is the "mouse," which is moved about the table surface to move the cursor about the screen.) these screen-oriented editors are generally accepted to be faster and more effective than the line-oriented editors, but they do require more elaborate equipment.

Somewhere between an information and idea organization tool and an interactive editing tool is the "structure" type of editor (17). A text editor works with normal text in terms of words, characters, and lines. A structure editor works with documents, sections, paragraphs, and so forth. The writer must specify a structure for his document, or take one which is already prepared on the system. The structure editor can help a writer work in a top-down manner from an outline. Sub-parts can be promoted, demoted, or rearranged in the structure.

After the text is structured and exists in the computer in a semi-finished form, there are still tasks the computer can perform for the writer. The computer can aid the writer by checking his spelling, checking his punctuation, and even checking his grammar. The Unix "Writer's Workbench" (18) is a computer system which contains facilities for many of these writing aids. Table 2 is a list of some of the "Writer's Workbenchf" commands with summaries of what they do.

An example of a text-manipulating computer system which fully exploits advanced screen-oriented techniques is the Xerox "8010 Star Information System" (19). This system organizes its screen like the top of an office desk. On this screen "desktop," the computer can display both sides of a regular-sized piece of typing paper simultaneously with a number of "icons." The icons each signify one of various operations that can be performed such as "delete a document," or "catalogue a document." The icon for "catalogue a document," for example, might be a picture of an office filing cabinet. The most meaningful icon may be the wastepaper receptacle which signifies deleting a document. The mouse is used for selecting spots on the pages for typing new or modified text as well as to pick

Table 2: Unix "Writer's Workbench" Commands.

ABST	Score text abstractness
ACRO	Find acronyms (words developed from the initial letters of names)
FINDBE	Identify awkward grammar
PARTS	Assign parts of speech
SEXIST	Locate sex-biased phrases
SPELLWB	Check spelling

icons to cause the computer to carry out operations.

Icons are a pictographic method for implementing "menus." A menu is a list of options presented to the user on the screen for selection. This is different from a command-driven interface in that the user's choices for the next operation are available for selection on the screen. In "menus" only the operations shown are allowed. It requires skill to design a menu-driven system to be certain that the operations required by the user at any point are available on the current menu (20).

The use of the screen to show how the finished document will look aids writers, especially where the arrangement and graphics are complex. The "desk-top" and "menu-driven" models for computer interaction are simple and easily learned by people who do not with to become adept at conventional computer command methods. This development will facilitate training for computer use in developing countries. For a complete survey of these interactive editing approaches see (21) and (22). These more effective user modes require what was once expensive computer equipment, but this capability is now made economically possible by the mass availability of personal computers.

A Computer-Based Personal Research and Writing Environment

Combining the above-described facilities on a modern, large-capacity personal computer results in a computer-based environment which is supportive of the work of an individual researcher/writer. This personal computer can be a "gateway" to bibliographic and full-text databases, allowing its user to draw material and references over communications lines from any database in the world. Tools can be brought to bear to analyze and organize this raw material of research. New material can be integrated into the researcher's own "knowledge graph" for his domain of interest or entered into his own bibliographic database (23). A structure editor can be used to synthesize research materials and ideas into new documents.

These functions can be accessed with a "desk-top" model screen display which allows more than once activity to be pursued at once. Bibliographic citations could be reviewed and classified and placed in an outline document structure at the same time. The weaving of these new materials into the researcher's "knowledge graph" could be interleaved with the other work. A capability for interleaving subtasks allows the researcher to proceed with his work in an order natural to the task and to the researcher. Features such as on-screen high-lighting and annotation which are special requirements of research and writing work could be included.

This system should be "open," that is, allowing the addition and subtraction of tools from the toolset according to the wishes of the individual user. The initial set of tools supplied with a copy of the system should be selected or developed to apply directly to the information retrieval and writing problem.

The Circle Closes

The researcher's new document can be added to the bank of knowledge through "electronic publishing." As documents were down-loaded, they can be up-loaded to bibliographic and full-text databases or transferred directly to other researchers, thus closing the research and writing circle.

The tools and methods outlined in this article exist in prototype or commercial versions on various personal computers, but are usually used in isolation. Efforts of the highest quality continue to integrate these tools into systems which not only accelerate but augment humans in doing knowledge work.

Research and writing are certainly a profitable area for the application of computers. It is to be anticipated that in the 1990's, perhaps the greatest progress in the application of computers will occur in these areas.

REFERENCES

[1] Lefrere, P., "Text Processing," in ARTIFICIAL INTELLIGENCE, edited by T. O'Shea and M. Eisenstadt, Harper and Row, New York, 1984.

[2] Lancaster, F., INFORMATION RETRIEVAL SYSTEMS, Wiley-Interscience, New York, 1979.

[3] Salton, G., "Automatic Information Retrieval," COMPUTER, September, 1980, p. 41-56.

[4] Glossbrenner, A., PERSONAL COMPUTER COMMUNICATIONS, St. Martin's Press, New York, 1983.

[5] Wente, V., "NASA/RECON and User Interface Considerations," in INTERACTIVE BIBLIOGRAPHIC SEARCH: THE USER/COMPUTER INTERFACE, edited by D. Walker, AFIPS Press, 1971, Montvale, New Jersey, p. 95-104.

[6] International Organization for Standardization (ISO), "Draft Specification for Commands for Interactive Search Systems," May, 1984, Document ISO/TC 46/4/5 N46.

[7] Marcus, R., "User Assistance in Bibliographic Retrieval Networks Through a Computer Intermediary," IEEE TRANSACTIONS ON SYSTEM, MAN, AND CYBERNETICS, Volume SMC-12, No.2,March/April 1982,p. 116-132.

[8] Maron, M., and Kuhns, J., "On Relevance: Probabilistic Indexing and Information Retrieval," JOURNAL OF ASSOCIATION FOR COMPUTING MACHINERY, Volume 7, 1960, p. 216-244.

[9] Luhn, H., "A Statistical Approach to Mechanized Encoding and Searching of Literary Information," IBM JOURNAL, October, 1957, p. 309-317.

[10] Salton, G., A THEORY OF INDEXING, Society for Industrial and Applied Mathematics, 1975.

[11] Bush, V., "As We May Think," ATLANTIC MONTHLY 176(7), p. 101-108.

[12] Price, L., "Thumb: An Interactive Tool for Accessing and Maintaining Text," IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, Volume SMC-12, No. 2, March/April 1982, p. 155-161.

[13] Wegner, P.,"Capital-Intensive Software Technology," COMPUTER, July, 1984, p. 4-45.

[14] Kaplan, S., "On the Difference Between Natural Language and High Level Query Languages," PROCEEDINGS OF THE ACM 78, Washington, D.C.

[15] Nau, D., "Expert Systems," COMPUTER, February, 1983, p. 63-85.

[16] Yip, Man-Kam, "An Expert System for Document Retrieval," M.S. Thesis in Electrical Engineering and Computer Science, MIT, Cambridge, MA., February, 1981.

[17] Walker, J., "The Document Editor: A Support Environment for Preparing Technical Documents," PROCEEDINGS OF THE ACM SIGOA SYMPOSIUM ON TEXT MANIPULATION, June 8-10, 1981, in SIGOA NEWSLETTER, Volume 2, No. 1 and 2, Spring/Summer, 1981, p. 44-50.

[18] MacDonald, N. Frase, L., and Keenan, S., "Writer's Workbench: Computer Programs for Text Editing and Assessment," Bell Laboratories, Piscateway, New Jersey, 1980.

[19] Smith, D., Irby, C., Kimball, R., Verplank, B., and Harslem, E., "Designing the Star User Interface," BYTE, Volume 7, No. 4, April, 1982, p. 242-282.

[20] Robertson, G., McCracken, D., and Newell, A., "The ZOG Approach to Man-Machine Communication," INTERNATIONAL JOURNAL OF MAN-MACHINE STUDIES, Volume 14, 1981, p. 461-488.

[21] Meyrowitz, N., and van Dam, A., "Interactive Editing Systems: Part I," COMPUTING SURVEYS, Volume 14, No. 3, September, 1982, p. 321-352.

[22] Meyrowitz, N., and van Dam, A., "Interactive Editing Systems: Part II," COMPUTING SURVEYS, Volume 14, No. 3, September, 1982, p. 353-415.

[23] Bertram,D., and Bader, C., "Storage and Retrieval of Bibliographic References Using a Microprocessor System," INTERNATIONAL JOURNAL OF BIO-MEDICAL COMPUTING, Volume 11, 1980, p. 285-293.

APPLYING NATURAL LANGUAGE TECHNIQUES TO IMPROVE MANAGEMENT INFORMATION SYSTEMS

Noemi M. Paz, University of Southern Mississippi
William E. Leigh, University of Southern Mississippi

Most managers have had little experience with "friendly" computer systems. They have traditionally been at the mercy of inflexible data processing systems which require the service of many programming specialists to supply special requests for information. Today there are natural language systems, mainly experimental, which are more responsive to the needs of the semi-trained or casual user.

Natural language as a man-machine communication method may be user "friendly" but is definitely not machine "friendly." Its ability to be vague and ambiguous has kept researchers from implementing an unrestrained natural language interface between humans and machine. Progress has been made in developing natural language question-answering systems. This paper is a survey of techniques that have been incorporated into natural language understanding so as to make the interfaces more amenable to human use, and which might be applied in the context of existing, non-natural language, management information systems.

INTRODUCTION

Natural language question-answering systems answer questions posed by a human user in the "natural language" (e.g., English) of the user. (See [8], [10], [13], [29], and [33] for descriptions of such systems.) These systems are limited to dialogues about restricted areas of an enterprise contained in a database, usually similar to a management information system. To a human user, the systems act as an interface between the user and the information in a much more user-oriented manner than the conventional, formal programming languages.

"If people from outside the computer fields are to be able to interact significantly with computers, then either they must learn the computer's languages or it must learn theirs" [31, p.183]. Learning the manager's language puts a burden on the computer system (contrast such an orientation to the usual inhuman and inhumane response such as "ERROR IER841I"). Not only must the machine understand the natural language but it must also tolerate simple errors, must embody a degree of "common sense," and must have a relatively large and complete management vocabulary.

In the broadest sense, there is a phenomenon that humans face sometimes called "computer shock" (see [26]). "The electronic revolution cannot be denied, but it can, perhaps, be understood and tamed, so that it serves human needs beyond the economic tally sheet." [26, p.339] Such sociological issues are not especially relevant to the function of a specific tool such as a question-answering system. However, one must deal with psychological resentment to a computerized system as implied by this notion. Alleviation of this situation is a primary goal.

Devices for ease of use center on removing the burden of communicating from the human and placing it on the machine. This is a main motivation for developing natural language capability.

Usually computer systems are designed with limited consideration of the behavioral implications of the human capabilities, limitations, and preferences. It is easier for the human designer to solve the closed problem of how to facilitate the computerization of a particular application instead of the open-ended problem of how the final product will interface with human users. "Invariably, most computer technicians approach systems from a very mechanistic stance. That is, they usually expect human systems to adjust to computer hardware and software." [27, p.297] As long as computer systems were used primarily for low-level tasks and by low-level personnel, this practice went uncorrected. However, today a question-answering system may be a component of a Decision Support System, which is used for strategic decision making by the higher levels of the organizational hierarchy. Such users have the clout to require a system that will be tailored exactly to their needs, style, and capabilities.

Certain techniques for accomodating human, user needs have been developed in natural language research which might be applied to systems which do not include natural language capability.

THE ENVIRONMENT OF A NATURAL LANGUAGE QUESTION-ANSWERING SYSTEM

A database contains tabularly organizaed information about a company's operation as compiled and maintained for the organization's accounting information system. A natural language question-answering system allows the posing of queries against that database in the natural language of the questioner. Examples of such queries include "How many refrigerators did John Jones sell last month?"; "List the sales of formula 19 last year in Sumatra"; and "What is the total value of spare parts for green Mustangs in the southeast region?"

Having user-type knowledge of the application domain is critical for the success of a natural language query system. This is different from knowledge of what data is used by management professionals. The knowledge needed is a concoptual view of the nature of the enterprise. This is independent of any data processing implications. The computerized end-user view of the enterprise contains descriptions of entities and their attributes, functions it can perform, and events which influence the way it performs.

Even though casual users understand the domain of discourse, their ability to pose reasonable questions against the stored knowledge may remain shallow. One reason this may come about is because they have assumed a particular structuring of the database which is not contained in the computerized, end-user view (the user has assumed non-existent relationships between entities). Some examples follow:

1. Presumption that timely data is maintained, as in a job cost database:

   Q: What is the current cost of work-in-progress for the Jonesville Dam Project?

2. Presumption of non-existent relationships, here incorrectly assuming cost is an attribute of plants:

   Q: What is the cost of all midwestern plants?

3. Assuming certain functions are applicable when in fact they are not. For example, if agricultural grading is recorded as characters A, B, and C, instead of numeric values, then the following is an incorrect functional use:

   Q: What is the average grade of our Spring wheat?

Only a general knowledge of the structure of the database can be attributed to the user by the question-answering system. The system must determine the appropriateness of the question to its database's contents and, if possible, return a suitable response to the user. This leads to the user posing questions that have "no" or "none" as answers. The null answer must be handled in such a way as not to seem curt.

Other issues arise in the presentation of the requested information in the manner intended by the user, in the graceful recovery from minor errors in input such as spelling, and in the resolution of ambiguities and imprecision inherent in the user's query. Not all question-answering systems try to handle each of these problems. Many of these systems are developed as part of current research in artificial intelligence and hence may focus on only one of these issues. However, the techniques developed in natural language research to deal with these problems can be adapted to handle the same problems in conventional information systems.

HANDLING THE NULL ANSWER

A "contributive" [3] response to a query which has the null answer would supply additional information which might be useful in the context of the original question. The answer "We have no information on x, but we do have 36 part number Y which is a substitute" is an example of this type of system behavior.

Questions of this type: "Is transportation cost included in overhead?" should be replied to with either a "yes" or with information about where transportation cost is really included. In general, the system should try to indicate the correct state of affairs rather than respond to such questions with merely a "no." [16, p.845].

The term "cooperative" [13] has come to signify this type of desirable behavior on the part of the system. This behavior recognizes the fact that the appropriate response may not be a direct answer to a question, but rather is an indirect, more informative, and perhaps in the human's perspective, a more correct response:

"Q: Which student got a grade of F in CS 105 in Spring '80?

A: None. (direct answer)

(alternate) A: CS 105 was not given in Spring '80." [13]

Clearly the issue faced here is that not only must the question be understood, but the machine must also have some understanding of the manager's intentions for asking the question. The computer can safely assume that the user does not ask idle questions and therefore it should provide information that the user is "obviously" missing. The computer now will appear to use "common sense" reasoning when responding to the human. Artificial intelligence "expert system" technology might be used to accomplish this for non-natural language information systems.

PRESENTATION OF THE REQUESTED INFORMATION

Decision Support Systems are characterized by a use of graphical output when possible. One natural language question-answering system in the literature follows this practice [PLANES (29)]. Otherwise, ouptup is usually organized in report formats.

Studies have been done on the impact of computer-generated drawings on humans. These studies have shown that aesthetic qualities found in hand-drawn graphs are important user-friendly interface aspects. For some applications computerized graphs are too mechanical and lack the warmth of hand-drawn lines. Algorithms have been developed that randomly disturb a regular pattern by small amounts and thereby give a hand-drawn quality which seems to enhance the saleability of the design. [19, p.7]

The earliest of question-answering systems [e.g. BASEBALL, (6 were designed to respond to ad hoc questions with natural, grammatical English sentences, but creating natural language is a step beyond understanding and this was never implemented. Instead systems have canned user friendly sentences to use when communication with the user is needed. Some systems (notably [3]) converse with the user to determine the intended layout of the output. The following is a sample of such a conversation:

USER: Give me a list of the part numbers, quantities, and suppliers of Houston parts.

SYSTEM: You are requesting (among other things) a listing of parts. Do you want:

1. quantities on hand
2. quantities on order
3. quantities shipped?

USER: I want qauantites on order.

SYSTEM: Fine! Now, with regard to "Houston parts," do you mean:

1. parts supplied by Houston suppliers
2. parts supplied to Houston projects
3. parts associated with Houston in some other way?

USER: I mean item 2.

SYSTEM: Good! I need on more piece of information. In the listing of part numbers, quantity on order, and suppliers, how do you want suppliers to be identified:

1. by number
2. by name?

USER: By name and by city.

SYSTEM: Do you want the listing to be ordered in any special way?

USER: Yes, by part number and then by supplier number [3, p.4].

These presentation techniques are certainly applicable to conventional, non-natural language information systems.

RECOVERY FROM INPUT ERRORS

A spelling corrector is a common component of natural language question-answering systems. (See [22], [16].) Other capabilities include the recognition of abbreviations [16] and facility for the user to define standard search arguments. The following example of a definition facility is from [3]. Consider the sample query: "Are any east coast suppliers supplying part 3g25?" Suppose the catalog of definitions does not include any definition for "east coast," "coast," or "east," and the lexicon does not include these words. The system can then ask the user whether "east coast" has anything to do with a supplier's rating, location, etc. The user would select "location," and the system would then ask the user to tell it which supplier locations are "east coast" [3, p.13]. This definition capability is a preventive measure for input errors.

Another system, LIFER [10] has the capability to extend the interface language to the personal needs or tastes of the human user. Initially the natural for the particular application area. A user, expert or novice, can extend the language by using the paraphrase facility which allows new grammatical structures to be defined in terms of old structures. For example: Let "describe part number 33621" be a paraphrase of "print the cost, color, and quantity on hand of part number 33621". The user could now expect the system to understand, for examply, "Describe the most expensive part." This user extensibility is also an attribute of some non-natural language information systems.

RESOLUTION OF AMBIGUITY

Further problems in understanding a natural language occur when the user enters an ambiguous statement. English is naturally ambiguous but many occurrences of ambiguity are resolved by searching the database for all the different, possible answers. If more than one possible answer is found, then the computer must ask the user to choose one of the possible meanings.

The dialogue supplied from [3] in "Presentation of the requested Information" (quoted presiously) is an example of a clarification dialogue for the resolution of vagueness in the query. However, many times input is vague or ambiguous because the user may engage in a dialogue with the computer. This naturally leads to inputs that are phrases and may refer back to previous questions.

The resolution of pronoun reference is a problem in this family. A record of the dialogue with the accountant is usually sotred as a type of "context register" [29] to enable the determination of the pronoun reference.

The "context register" is also useful in determining the intent of "ellipsis," i.e., missing sentence constituents. "Case frames" [29] are a complementary technique for handling this problem. "Case frames" establish scripts of expectation to furnish defaults for the missing elements of the query. This, of course, makes various assumptions about the user's intent.

Abbreviation is another type of input form that is allowed by many of these systems in an effort to lessen the amount of typing the user must do. Abbreviations are placed into the system's dictionary so that they can be recognized. Problems with ambiguous input arise, for example, in resolving whether a user meant the preposition or the state Indiana when "IN" is entered as part of a question.

USER EDUCATION TECHNIQUES

The area of improving human performance and usability of database systems is large and has many fronts. However, it can be divided into two main categories:

1. What can be done "to the computer" to make it easier to use?
2. What can be done "to the human" to make the computer easier for him to use?

We have discussed techniques that can be included in the computer programs to enhance the human factors aspect of database interactions. This section concerns the second category.

One step is to study the human problem-solving process. This does not necessarily mean studying cognitive thought processes, but may involve studying the methodology that is followed and what tools are used. Malhotra [16] developed an experiment to study managers as they solved problems. He used his experiment to discover how managers reach an understanding of a problem situation by studying the questions made to a database system with a natural language interface.

A manager participating in Malhotra's experiment had the assistance of a "Perfect System" that would respond to any natural language request made to it. Answers were communicated to the subject through a computer terminal by a human.

One result of Malhotra's study was a determination of requirements for a "Perfect System." The list of requirements included facilities for data retrieval and manipulation, for answering questions about the data it maintained and its capabilities, and for building and using management models. The necessary capabilities for a natural language interface were also defined.

Given current technology, the "Perfect System" cannot yet be built. In a real information system not all questions can receive a response, thus reducing the system's ease of use and increasing the frustration of a human user of the computer. Perhaps there is a way of changing problem-solving behavior so that the feasible system is perceived to be as successfully friendly as the "Perfect System."

Mayer [17] has conducted a series of studies that show that concrete computer models enhance performance when given to a subject prior to learning how to use a system. His study revealed that there were more positive results for novices than for subjects who already had technical knowledge of the system.

In Mayer's studies half of the subjects were given a concrete model of the computer for a BASIC-like language. Then all the subjects read a manual on the BASIC-like language and took a test on the material. Results showed that those given the model prior to reading the manual were able to use the model as an underlying framework for understanding the new information.

A casual user of a database system usually has little knowledge of the computer or the organization of the data. With this limited knowledge this user still needs to do careful thinking in order to pose questions that can be answered by the computer. Studying a concrete model before or while being introduced to the natural language question-answering system may improve the percentage of questions that can be successfully answered. This casual type of user, as suggested by Mayer [17], may be the one most helped by such a model.

Preparing the user in this way can reduce the number of unanswerable questions he poses. Such pre-conditioning can correct such sources for error as:

1. User has unrealistic expectations in the power of the computer. Questions that expect the machine to make a value judgment fail under this type.
2. User requests information not contained in the database.
3. Questions cannot be answered because of the way the data is structured.
4. Ambiguous queries need clarifying dialogue for resolution.
5. Questions need specialized management models or mathematical functions that are not implemented in the system.

Implicit in number three above is that the structure of the data is somehow included in the concrete model. One area of possible research is to evaluate the effectiveness of using the relational, hierarchical, or network data model to explain structure of data to the casual user. Perhaps experimentation will show that no one model is best, or that the best one for the situation depends on the application area, the management position of the casual user, and the previous computer experience of the user.

SUMMARY AND RECOMMENDATIONS

Techniques described for giving computers "interpersonal" skills in the database question-answering situation were cooperative responses, recovery from minor input errors, varied presentation of requested information, and extensibility of the natural language understood by the computer. These techniques can be applied to conventional, non-natural language systems.

Further work in this area should attempt to evaluate the worth of these measures. Each of the features mentioned above should be evaluated to determine if they are, in fact, improving the user's satisfaction with the system. "Quality as perceived and defined by the user is (and always should be) the dominant evaluation criterion in the effective design of information systems." [4]

A very interesting feature used to some extent by Codd [3] is "stroking" behavior on the part of the question-answering system. (See the "Good's" in the dialogue in the "Presentation..." section of this paper.) Such a feature might be economically included in systems without natural language capability.

The behavioral emphasis in natural language question-answering systems has been toward creating a computer system with user-friendly elements. Developing a concrete model for users to receive prior to using the computer system is an alternate strategy. The hope is that the right model will provide the user with a helpful mental model of the computer system and its capabilities. This model could be tailored to managers as the specific "users" of interest.

REFERENCES

[1] Barr, A., and Feigenbaum, E. (ed.), THE HANDBOOK OF ARTIFICIAL INTELLIGENCE,Vol.I (Los Altos, California: William Kaufmann, Inc., 1981).

[2] Buckley, J., ACCOUNTING, AN INFORMATION SYSTEMS APPROACH (California: Dickenson Publishing Co., 1973).

[3] Codd, E., "Seven Steps to Rendezvous With the Casual User," in J. W. Klimbie and K. L. Koffeman (eds.).,DATA BASE MANAGEMENT (Amsterdam, North-Holland, 1974), pp. 179-200.

[4] Elam, P., "User-defined Information System Quality," JOURNAL OF SYSTEMS MANAGEMENT, Vol. 30, No. 8 (August 1979), pp. 30-33.

[5] Embley, D., and Nagy, G., "Behavioral Aspects of Text Editors," ACM COMPUTING SURVEYS, Vol. 13, No. 1 (March 1981), pp. 33-70.

[6] Green, B., Wolf, A., Chomsky, C., and Laughery, K., "BASE-BALL, An Automatic Question Answerer," in Feigenbaum and Feldman (1963), pp. 207-216.

[7] Harris, L., "User Oriented Data Base Query with the ROBOT Natural Language query System," INT. JOURNAL MAN-MACHINE STUDIES, Vol. 9 (1977), pp. 697-713.

[8] Harris, L., "Experience with ROBOT in Twelve Commercial Natural Language Data Base Query Applications," IJCAI, 6 (1979), pp. 365-368.

[9] Hayes, P., Ball, E., and Reddy, R., "Breaking the Man-Machine Communication Barrier," COMPUTER, Vol. 14, pp. 319-330.

[10] Hendrix, G., "Human Engineering for Applied Natural Language Processing," IJCAI, 5 (1977), pp. 183-191.

[11] Hendrix, G., Sacerdoti, E., Sagalowicz, D., and Slocum, J., "Developing a Natural Language Interface to Complex Data," ACM TRANSACTIONS ON DATABASE SYSTEMS, 3 (1978), pp. 105-147.

[12] Jastrow, R., "Toward an Intelligence Beyond Man's," in H. Watson and A. Carroll (eds.), COMPUTERS FOR BUSINESS (1980), pp. 340-342.

[13] Kaplan, S., "Cooperative Responses from a Portable Natural Language Data Base Query System." Doctoral dissertation, Department of Computer and Information Science, University of Pennsylvania, 1979.

[14] Keen, P., "Information Systems and Organizational Change," CACM, Vol. 24, No. 1 (January 1981), pp. 24-33.

[15] Ledgard, H., Whiteside, J., Singer, A., and Seymour, W., "The Natural Language of Interactive Systems," CACM, Vol. 23, No. 10 (October 1980), pp. 556-563.

[16] Malhotra, A., "Knowledge-Based English Language Systems for Management Support: An Analysis of Requirements," IJCAI (Tbilisi, USSR: 1975).j

[17] Mayer, R., "The Psychology of How Novices Learn Computer Programming," ACM COMPUTING SURVEYS, Vol. 13, No. 1 (March 1981), pp. 121-141.

[18] Mays, E., "Failures in Natural Language Systems: Applications to Data Base Query Systems," PROCEEDINGS OF THE FIRST ANNUAL NATIONAL CONFERENCE ON AI (Stanford University: August 1980), pp. 327-330.

[19] Myers, W., "Computer Graphics: Reaching the User," COMPUTER, Vol. 15, No. 3 (March 1981), pp. 7-17.

[20] Porter, W. T., EDP CONTROLS AND AUDITING (California: Wadsworth Publishing Co., 1974).

[21] Reisner, P., "Human Factors Studies of Database Query Languages: A Survey and Assessment," ACM COMPUTING SURVEYS, Vol. 13, No. 1 (March 1981), pp. 13-32.

[22] Sacerdoti, E., "Language Access to Distributed Data with Error Recovery," IJCAI, 5 (1977), pp. 196-202.

[23] Shneiderman, B., "Improving the Human Factors Aspect of Database Interactions," ACM TRANS ON DATABASE SYSTEMS, Vol. 13, No. 4 (December 1978), pp. 417-439.

[24] Siklossy, L., "Impertinent Question-answering Systems: Justification and Theory," ACM PROCEEDINGS ANNUAL CONFERENCE (Washington, D.C., 1978), Vol. 1 pp. 39-44.

[25] Sprague, R., and Watson, H. "A Decision Support System for Banks," in COMPUTERS FOR BUSINESS, H. Watson and A. Carroll (eds.) (1980), pp. 211-226.

[26] Stewart, J., "Computer Shock: The Inhuman Office of the Future," in COMPUTERS FOR BUSINESS, H. Watson and A. Carroll (eds.) (1980), pp. 332-339.

[27] Tomeski, E., Stephenson, G., and Man Yoon, B., "Behavioral Issues and the Computer," in COMPUTERS FOR BUSINESS, H. Watson and A. Carroll (eds.) (1980), pp. 294-301.

[28] Vyssotsky, V., "Computer System in Business: More Evolution than Revolution," MANAGEMENT REVIEW (February 1979), pp. 15-22.

[29] Waltz, O., "An English Language Question-Answering System for a Large Relational Database," CACM, Vol. 21, No. 7 (July 1978), pp. 526-539.

[30] Weizenbaum, J., "Csontextual Understanding by Computers," CACM, Vol. 10, No. 8 (August 1967), pp. 474-480.

[31] Weizenbaum, J., COMPUTER POWER AND HUMAN REASON (San Francisco: Freeman and Co., 1976).

[32] Winograd, T., "When Will Computers Understand People?" PSYCHOLOGY TODAY (May 1974), pp. 73-79.

[33] Woods, W., "Syntax, Semantics, and Speech" in R. Reddy (ed.), SPEECH RECOGNITION: INVITED PAPERS OF IEEE SYMPOSIUM (New York: Academic Press, 1975), pp. 345-400.

SOFTWARE SUPPORT SYSTEMS FOR USE IN TECHNOLOGY TRANSFER ACTIVITIES

G. David Huffman**

ABSTRACT

Technology transfer can be carried out in a formal manner by technology transfer agents or or an informal basis by technology generators or recipients. A number of modes of technology transfer are now employed by technology transfer agents which principally vary in degree of specificity. These modes are largely the same regardless of the technology under consideration. The processes are labor intensive and are now being automated using micro-computer based techniques and software support systems. The software support systems encompass pre-processors for database access, gateway systems, post-processors for citation analysis, databases on technology experts, report organization and writing, text processing, communications, statistical analysis and program integration. Much of the above software is commercially available with a few programs under development. In particular, a citation analysis program using lexical association methods is in the prototype

---

*This study was supported in part by the Technology Utilization Office, NASA, through a subcontract from the Indianapolis Center for Advanced Research.
**Dr. Huffman is the Dean of the College of Science and Technology and a Professor of Computer Science at the University of Southern Mississippi, Hattiesburg, Mississippi.

commercial and/or public domain modules is an ever present

stage and should significantly improve on-line database search precision without reducing search breadth. Integration of the problem with federally funded activities underway in this area. All in all, the full implementation of the previously mentioned software systems can reduce technology transfer study costs by up to 15%.

INTRODUCTION

Technology transfer can be carried out in a formal manner, i.e., by third-party organizations such as technology transfer agents, or in an informal manner within or external to the technology generating or receiving organization. Typical modes of formal technology transfer can be categorized as: current awareness searches, retrospective searches, technology assistance programs and industrial applications studies. These modes differ in specificity but all include the location of existing technology which is relevant to a specific problem. The technical assistance program and the industrial applications study also encompass the reformulation of the technology to a form which is understandable to and useable by the technology recipient.

The steps utilized in an industrial applications study are shown in Figure 1 and consist of locating the technology, analyzing bibliographic information, contacting experts and formulating a final report. The technology acquisition process consists of carrying out on-line database searches and locating documents which are relevant to the problem at hand. Since the processes utilized in technology transfer activities of this type are the same regardless of the technology, considerable labor savings can be realized by automating the processes.

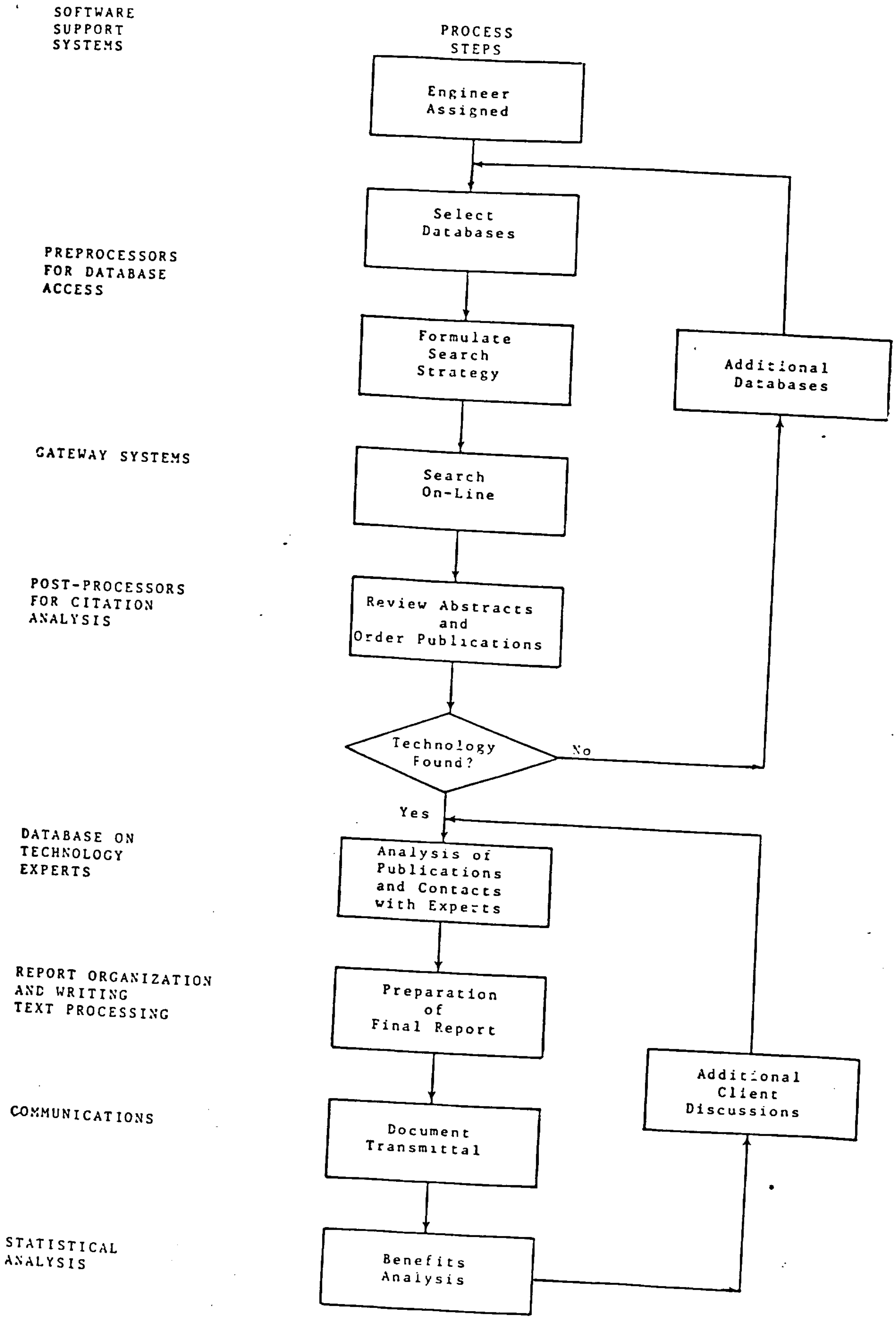


Figure I. Profile of Steps Carried Out in an Industrial Applications Study

Prior to discussing the various software support systems and the automation of the process, it is useful to review the manual mode of operation. Given a well defined problem statement, the technology transfer professional selects a number of databases, familiarizes himself with the database thesauri, selects a series of keywords and searches the bibliographic databases on line. Hardcopies of the abstracts are then requested and--upon receipt--are analyzed. At this point, the search can be reinitiated if the original results failed to locate relevant technology. Not that the analysis process is time-consuming in that the most thoroughly formulated search strategy rarely yields more than about 10% relevance abstracts.

Following an analysis of the abstracts, documents are ordered and subsequently analyzed. Additional expert assistance may be utilized and the client may be contacted for additional data. Following this technology reformulation step, the final report is prepared and transmitted to the client. Feedback from the client is solicited and can be used to determine the efficacy of the study.

The process described in the preceding paragraphs can be automated using the following scenario. A disk file is established on a mini- or micro-computer for a specific study. The external databases are searched and abstracts are then evaluated for relevancy either manually and/or by a lexical association software system. Document order forms are prepared by the software system, and the relevant abstracts retained on the disk file. Documents are again evaluated, industrial contacts made, and the final report is prepared. The final report preparation employs word processing with the text and abstract files merged. The final document can then be printed or transmitted directly to the sponsor over a network or phone line. Finally, the benefit analysis is merged with the project file.

The software support systems to be employed in the task automation are shown in Figure 1. The various modules consist of: pre-processors for database access, gateway systems, post-processors for citation analysis, a technology expert database, report organization and text processing software, communications software and statistical packages.

The technology transfer process from a software point-of-view can be approached from a concentrated or distributed perspective. The concentrated concept places all software on one fairly large computer, i.e., a gateway, with database access and all other functions on the same machine. Users access this computer over telephone lines or a communication network. The approach advocated in this work follows a distributed perspective with the computing resources located at the various sites carrying out the technology transfer studies. Computer equipment requirements are substantially reduced, i.e., a micro- rather than a mini-computer, since the system supports only one user at a time.

The reasons for the choice of a distributed approach are:

i. The rapid increase in micro-computer capability. A number of manufacturers are now marketing micro-computers having 3 mb of main memory and 40 mb of fixed disk memory in desk-top units.

ii. Cost trends. Micro-computer costs have fallen when compared to communications costs. This factor favors distributed computing over a central, i.e., gateway, computer.

iii. The availability of micro-computer software. There is no question that the number of technology transfer oriented computer programs has increased many-fold in the last year.

As a consequence, all the software systems discussed in the following section are operable on a suitably configured micro-computer.

SOFTWARE TOOLS

The software support systems described in the ensuing section are all operable on a 16-bit micro-computer utilizing an MS-DOS operating system. This is exemplified by an IBM PC and the minimum computing environment is denoted as an "IBM PC and compatibles".

Commercial and government developed software packages corresponding to the SOFTWARE SUPPORT SYSTEMS of Figure 1 are shown in Figure 2. TABLE 1 gives a brief description of the functionality of the program along with the minimum operating environment. The software support systems shown are all inclusive in that they will provide sufficient capabilities to fully automate the various tasks of Figure 1. The list does not, however, attempt to delineate all possible commercial systems within a functional area. For example, there are probably 50 text processing programs available and a comprehensive list would consume numerous pages. The listed programs should be viewed as typical and capable of supporting the required functionality rather than comprehensive. The user friendliness, documentation, etc., varies substantially from one program to another. Since the assessment of program efficacy is frequently a matter of personal preference, no value judgments are being offered with regard to the listed programs--"What's one man's poison is another's meat or drink". Prices on software systems are frequently determined by dealer's and educational discounts. As a consequence, pricing information is not included; however, vendors, addreses, and phone numbers for all the listed software systems are included in the REFERENCES.

SOFTWARE SYSTEMS	FUNCTIONAL AREA
IN-SEARCH MIST + PCCL (1)	Pre-processors for database access
Business Computer Network	Gateway Systems
SORT-AID (1) NUTSHELL	Post-Processors for Citation Analysis
dBASE II	Database programs
THINKTANK OFFIX	Report Organization
WORDSTAR SELECT WRITE	Text Processing
CONEXUS	Communications
STAT-PAK	Statistical Analysis
MEMORY/SHIFT	Integration

----

(1) Under development at the University of Southern Mississippi

Figure 2. Typical Software Packages for Use in Technology Transfer

TABLE 1
DESCRIPTIONS OF SOME SOFTWARE SYSTEMS RELEVANT TO TECHNOLOGY TRANSFER TASKS

Program Title	Description	Environment
IN-SEARCH	Front-end database searching program. Catalog of all DIALOG databases and resources. Search revision capabilities and keyword high-lighting within each reference.	IBM PC and compatibles. 192 K; two disk drives; smart or acoustic modem.
MIST +	Microcomputer communications program. Program contains a full programming language with specifications for telecommunicating, a database system and a text editor. The database can be turned into a full-fledged computer teleconferencing system complete with electronic mail, conferences and on-line databanks.	IBM PC and compatibles. 256 K; two disk drives; hard disk recommended; smart or acoustic modem.
PCCL	Menu driven program which allows formulation of a series of command languages.	IBM PC and compatibles. 128 K; one disk drive; smart or acoustic modem.
Business Computer Network	Logs on automatically to a number of on-line information systems. Program captures text on disk, writes messages off-line, sends them on-line and sends sequences to a printer.	IBM PC and compatibles. 128 K; two disk drives; Hayes compatible modem.

TABLE 1 (continued)
DESCRIPTIONS OF SOME SOFTWARE SYSTEMS RELEVANT TO TECHNOLOGY TRANSFER TASKS

Program Title	Description	Environment
SORT-AID	Uses lexical association methods and artificial intelligence frames to rank abstracts by relevance.	Under development for operation on an IBM PC/XT. Currently operational on a DEC VAX 11/780 with the VMS 4.0 operating system.
NUTSHELL	Citation analysis system which allows review, categorization, etc. Records can be indexed by title, author, keywords, etc.	IBM PC or compatibles. 128 K; one disk drive; smart or acoustic modem.
dBASE II	Database program which constructs and manipulates numeric and character data. Provides database manipulation directly from a keyboard. Provides capability for user generated menus and application programs.	IBM PC or compatibles. 256 K; two disk drives.
THINKTANK	Text processing program oriented toward report organization. Uses outlining technique.	IBM PC/XT or compatibles. 256 K; one disk drive.
OFFIX	Personal office system which mimics a file cabinet. Software can search a datafile for up to 10 fields simultaneously and then sort by one of the ten. Can also send information to a screen or printer.	IBM PC/XT or compatibles. 256 K; one disk drive.

TABLE 1 (continued)
DESCRIPTIONS OF SOME SOFTWARE
SYSTEMS RELEVANT TO TECHNOLOGY
TRANSFER TASKS

Program Title	Description	Environment
WORDSTAR	Screen oriented word processing system featuring integrated printing. Displays both initial entry of text and alteration of previously entered text. Includes features for automatic margins, justification and paging.	IBM PC or compatibles. 128 K; one disk drive.
SELECT WRITE	Screen oriented word processing system featuring key files for storing frequently used paragraphs and single key insertion into a document. Includes features for automatic margins, justification and paging.	IBM PC or compatibles. 128 K; two disk drives.
CONEXUS	Communications system providing electronic mail, bulletin boards, teleconferencing, etc. Includes a password access system.	IBM PC/XT or compatibles. 256 K; two disk drives; smart modem.
STAT-PAK	Performs univariate analysis of data. Includes descriptive statistics, correlation, linear regression, analysis of variance, analysis of covariance, goodness of fit and test of independence.	IBM PC or compatibles. 128 K; one disk drive.
MEMORY/SHIFT	Provides capability for simultaneous operation of programs and inter-program data transfers.	IBM PC/XT or compatibles. 128 k; one disk drive.

Minimum equipment requirement. The software vendors are listed in the REFERENCES. 128 K denotes 128kb of main memory. IBM PC/XT or compatibles denotes micro-computer having 10MB fixed disk drive. One disk drive denotes 320-360kb removable disk drive, i.e., floppy disk drive.

UNRESOLVED ISSUES

While the programs of Figure 2 and Table 1 are functionally complete, two issues remain to be resolved--one key program is still under development and the software systems are configured to operate in the stand-alone mode. As noted in Huffman and Leigh (1984), the review of abstracts for relevance to the problem statement is a very labor intensive task with 80-90% of the captured abstracts frequently discarded. Consequently, methods to improve search relevance without reducing breadth of coverage can produce significant labor and cost savings. The program SORT-AID has been designed to carry-out this task. SORT-AID is being developed at the University of Southern Mississippi under NASA sponsorship, Huffman and Leigh (1984).

The SORT-AID program uses a method for semi-automatic relevance ranking based on lexical association, Salton (1975). The index theory is applied to the abstracts which result from a database search. The abstract collection is, thus, homogeneous which accentuates the characteristics of the indexing methods.

The relevance ranking is similar to keyword querying with the keywords selected from lists generated by the indexing algorithms applied to the post-search abstract collection. Two lists of keywords are generated by SORT-AID. The first is based on collection frequency and contains the thirty most frequently used words in the abstracts exclusive of words such as a, an, the, etc. The second set of keywords is based on the signal-to-noise ratio, i.e., a parameter characterizing the global frequency of occurence divided by a parameter characterizing a global background frequency. The user selects words from the two lists which are relevant to the specific problem. The relevance score for each citation is then computed using the above user supplied keyword relevancy. The abstracts are then presented to the searcher in order of relevance and he can review a fixed number or all of the citations.

As noted above, the software systems of Figure 2 are independent programs and, thus, have different file structures, data formats, etc. This implies that all the programs must function independently. This is an inconvenient operating mode and could be substantially improved by integrating a selected subset of the programs. MEMORY/SHIFT will provide limited capability to integrate programs. Further program integration is required for a functional "searcher's workbench".

SUMMARY

Technology transfer activities can be carried out in a formal or informal manner. Technology transfer agents employ the formal approach which consists of locating the relevant technology and reformulating it to a form which is usable by the recipient. The mechanisms employed in the technology transfer activities are more or less the same regardless of the technology. Furthermore, the activities are labor intensive. Significant manhour savings can be realized by automating--via computer systems--the technology transfer processes. Software systems are currently available or under development which can yield significant cost and time reductions and are described in this paper.

REFERENCES

[1] Business Computer Network, P. O. Box 36, 1000 College View Drive, Riverton, WY 82501, 800-446-6255

[2] CONEXUS, New Era Technologies, Inc., 1252 Columbia Road NW, Washington, DC 20009, 202-887-5440

[3] dBASE II, Ashton-Tate, 10150 W. Jefferson Blvd., Culver City, CA 90230, 213-204-5570

[4] Huffman, G. David and Leigh, W. "First Quarterly Report," NASA Contract Number NASW-3593-modification Number 6, November, 1984

[5] IN-SEARCH, Menlo Corporation, 4633 Old Ironsides - Suite 400, Santa Clara, CA 95050, 408-986-0200

[6] MEMORY/SHIFT, North American Business Systems, 642 Office Parkway, St. Louis, MO 63141, 800-325-1485

[7] MIST+, New Era Technologies, 1252 Columbia Road NW, Washington, DC 20009, 202-887-5440

[8] NUTSHELL,Leading Edge Products, 21 Highland Circle, Needham, MA 02194, 800-343-3436

[9] OFFIX, Emerging Technology Consultants, Inc., 1877 Broadway Boulder, CO 80302, 303-447-9495

[10] PCCL, University of Southern Mississippi, College of Science and Technology, Southern Station Box 5165, Hattiesburg, MS 39406-5165, 601-266-4883

[11] Salton, G. "A Theory of Indexing," Society for Industrial and Applied Mathematics, Philadelphia, PA, 1975

[12] SELECT WRITE, Select Information Systems, 919 Sir Francis Drake Blvd., Kentfield, CA 94904, 415-459-4003

[13] SORT-AID, University of Southern Mississippi, College of Science and Technology, Southern Station Box 5165, Hattiesburg, MS 39406-5165, 601-266-4883

[14] STAT-PAK, Science Software, RFD2 Box 63, Nelsonville, OH 45764, 614-753-1397

[15] THINKTANK, Living Videotext, Inc., 2532 Charleston Road, Mountain View, CA 94043, 415-964-6300

[16] WORDSTAR, Micropro International Corporation, 33 San Pablo Avenue, San Rafael, CA 94903, 415-499-1200

A PC-BASED GATEWAY FOR INFORMATION RETRIEVAL

William Leigh
Univ. of Southern Mississippi, Hattiesburg, Mississippi, U.S.A.

Ray Souder
Northern Kentucky Univ., Highland Heights, Kentucky, U.S.A.

There are many bibliographic citation databases available to computer terminal equipment over telephone lines. These databases are useful for technological research and the development of bibliographies.

This paper describes the design and use of a user-interface for these information retrieval systems which achieves a common command language for several databases. This user-interface resides in a personal-computer and mediates the researcher's dialog with the online systems.

INTRODUCTION

Since the 1960's, when the enabling computer technology became available, many bibliographic citation and full-text databases covering almost any area of knowledge have been built and made available for access with computer terminal equipment via telephone lines. Figure 1 is a diagram of this arrangement. Lancaster (1979) supplies a good overview of the history and capabilities of information retrieval systems. Salton (1980) offers a survey of technical developments in the field.

Bibliographic databases contain citation entries composed of titles, authors, publication information, and abstracts of articles, books, and reports. Full-text databases contain the complete texts of the documents in addition. There is one entry in these databases for each document. Usually the databases specialize in one area of knowledge. Glossbrenner (1983) is a directory to the most widely available databases, what they contain, and how they are accessed. Table 1 contains a list of some major databases and a description of their contents. Wente (1971) describes one bibliographic database system in detail.

Entries are accessed with keywords. Keywords are supplied for the documents by author, by a human indexer, or with automatic methods. Figure 2 contains a citation including keywords which might be from a computer science bibliographic database.

Queries are formulated by the researcher to access databases. These queries must be expressed in the special language supported by the specific database. Efforts are underway to standardize these languages across databases. The International Organization for Standardization (ISO) is

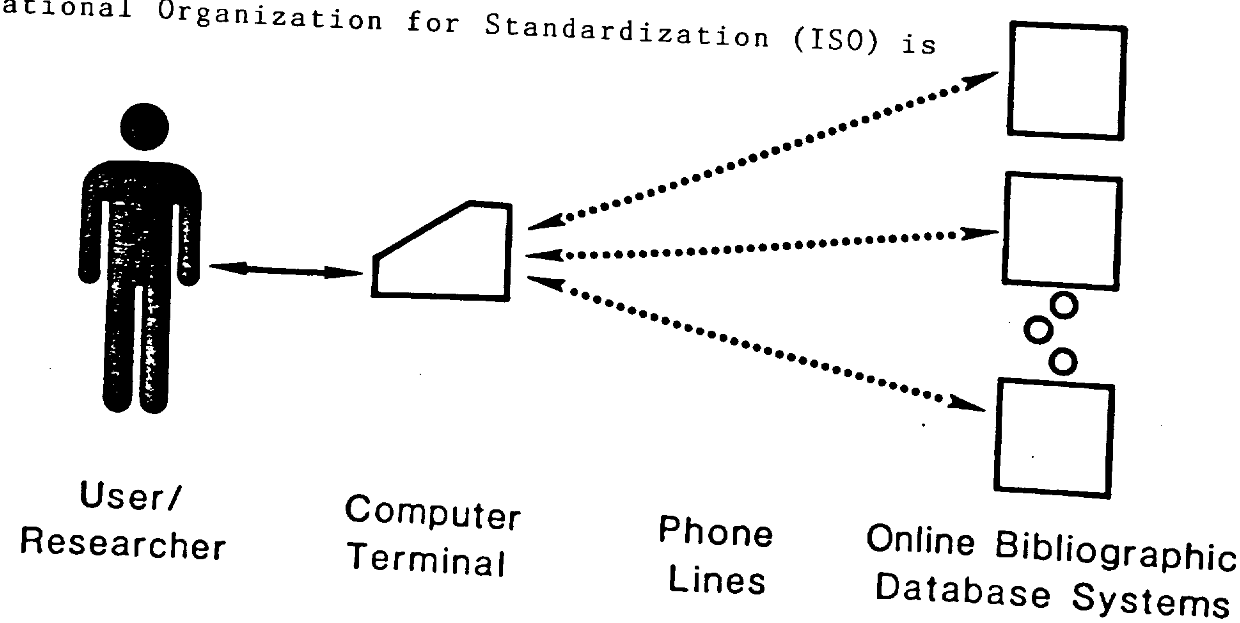


Figure 1
ONLINE INFORMATION RETREIVAL

ABI/INFORM	Business and Economics
AGRICOLA	Agriculture and Related Subjects
COMPENDEX	Worldwide Coverage of Engineering
ERIC	Educational Materials
INSPEC	Physics, Electrotecnology, Computers
MEDLINE	Medicine, Dentistry, Nursing
NASA/RECON	Aerospace Technology

Table 1
Some Major Databases and Their Subjects

Title	-	A Data Structure for Word Processing
Author	-	Jones, John M.
Date	-	August, 1982
Source	-	Journal of Computer Science
Pages	-	354-368
Keywords	-	DATASTRUCTURES,WORDPROCESSING, OFFICE AUTOMATION, TEXT PROCESSING, TEXT EDITING
Abstract	-	A data structure is presented which has desirable properties in situations requiring automatic justification under conditions of insert and overwrite. This situation occurs in screen-oriented word-processing systems. Examples of use and test results are reported.

Figure 2
Citation from a Typical Bibliographic Database

considering one such standard language as described in the "Draft Specification for Commands for Interactive Search Systems" (1984). Figure 3 contains several queries expressed in this draft standard query language. Another solution to the problem of multiple querying languages which does not involve changing the database systems themselves is the "gateway" approach. Marcus (1982) has built such a system which uses a computer to convert queries specified in a standard language to the specific languages of the individual database systems. Figure 4 shows the relationship of the components in the "gateway" approach to online information retrieval.

```
FINE WORD PROCESSING AND DATA STRUCTURES
FIND (WORD PROCESSING OR TEXT PROCESSING) AND TEXT EDITING
FIND (DATA STRUCTURES OR ALGORITHMS) AND WORD PROCESSING
FIND TEXT EDITING AND DATE 1977
```

Figure 3
Example ISO Standard Queries

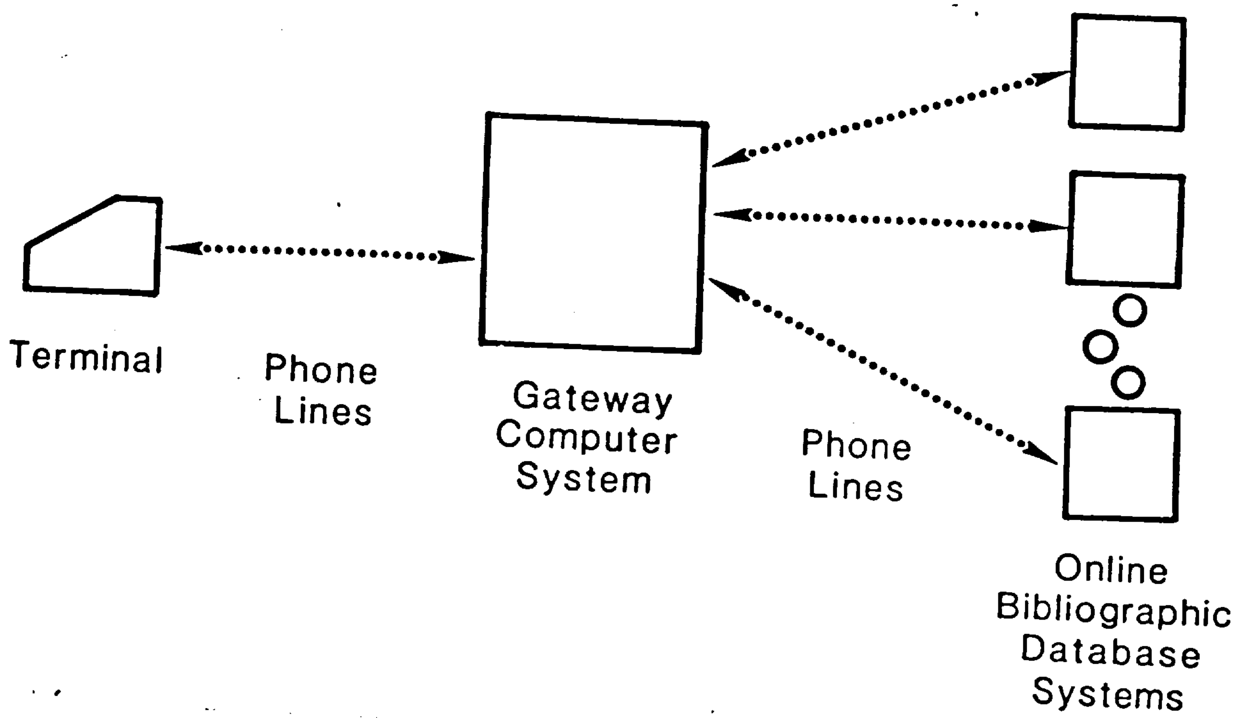


Figure 4
DIAGRAM OF A "GATEWAY" ARCHITECTURE

DEVELOPING A GATEWAY

The development of a "gateway" is a difficult task. There are many subtleties of using the various databases that are not apparent in the manuals. Developing truly equivalent commands which remain equivalent across several databases may not be possible. However, before searching can be done by non-experts, this must be accomplished.

For example, a first issue to be resolved is the meaning of "basic index". The different databases have different semantics for this term. It is logical that the user be presented with the same basic index on every database. The basic index which makes the most sense includes all searchable terms and stems from title, abstract, and the controlled vocabulary of the system.

This default index should be set up by the "gateway" for each system as the default by the logon--this process being "transparent to the user."

The semantics of the search language for the various systems is often obscure. For example, a single term search template for NASA/RECON:

```
SEARCH searchterm + ATL/searchterm + AX/searchterm
```

An AND search for term 1 AND term 2:

```
SEARCH (searchterm1 + ATL/searchterm1 + AX/searchterm1)
       *(searchterm2 + ALT/searchterm2 + AX/searchterm2)
```

A MENU-STYLE INTERFACE

Our system is based on the ZOG(7) approach to non-machine communication. A network of menu's is presented to the user. That network is designed by an expert in the application domain, in this case computer bibliographic searching.

Indented menu's are available for several databases. (See Figure 5 and 6 for example) The user can select a function with confidence that semantically equivalent commands are transmitted to the subject database system.

```
        MASTER MENU

A     DROLS MENU
B     NASA/RECON MENU
C     EUROPEAN STANDARD MENU
D     POST-PROCESSING MENU
E     MODIFY PCCL
G     PRINT SEARCH TERMS
H     ENTER/CHANGE SEARCH TERMS
X     RETURN TO SYSTEM

--enter your selection

                                        F1 - help, F2 - end
```

Figure 5
The Master Menu

```
        EURPOEAN STANDARD MENU

A         SIGNON
B         HELP
C         SELECT DATABASE
D         TERMINATE SESSION
E         DISPLAY SEARCH TERMS
F         SEARCH ST1
G         SEARCH ST2
H         SEARCH ST1 AND ST2
I         DOWNLOAD CITATIONS
J         SPECIAL FEATURE
K         ENTER/CHANGE SEARCH TERMS
X         RETURN TO PCCL MASTER MENU

--enter your selection

                                        F1 - help, F2 - end
```

Figure 6
An example of a search menu

This menu-style interface is implemented on a personal computer. The menu's and their underlying commands may be coded by the experts or by the user with a screen-oriented text editor. Also, expanded HELP texts may be prepared.

In operation, the user can employ his own communication program to handle the telephone lines between his personal computer and the remote online database system. A commercial "windowing" environment program is used to accomodate the communication program and our menu-style interface simulataneously on the same personal computer. The user can activate our program and select his function. The underlying data base commands may then be passed to the communication program with the Facilities of the "windowing" program and then transmitted to the remote database system.

SUMMARY

There are many bibliographic citation databases available to computer terminal equipment over telephone lines. These databases are useful for technological research and the development of bibliographics. Usually, for a researcher to use these databases, he must seek the assistance of an "expert intermediary" who specializes in the details on online searching and knows the command languages of the different systems.

We have built a user-interface for these information retrieval systems which achieves a common command language for several databases. This user-interface resides in a personal computer which acts as a "gateway" to the bibliographic database systems. This user-interface mediates the dialog between the user and the online systems. This system functions in a menu-based manner. Integral help facilities are present. User extension is allowed.

This interface is a "first-aid kit". Lowering of the syntax barrier makes searching a do-it-yourself proposition. Many engineers wish to be self-sufficient and not dependent an expert. This tool relieves the expert of syntax problems, also. Most importantly, this system provides a convenient tool for structuring and codifying expert searcher's knowledge.

REFERENCES

[1] Glossbernner, A., Personal Computer Communications (St. Martin's Press, New York, 1983).

[2] International Organization for Standardization (ISO), Draft Specification for Commands for Interactive Search Systems (Document ISO/TC 46/4/5 N49, 1984).

[3] Lancaster, F., Information Retrieval Systems (Wiley-Interscience, New York, 1979).

[4] Marcus, R., User Assistance in Bibliographic Retrieval Networks Through a Computer Intermediary, IEEE Transactions on System, Man, and Cybernetics, Vol. SMC-12, No. 2 March/April (1982) 116-132.

[5] Robertson, G., McCracken, D., and Newell, A., The ZOG Approach to Man-Machine Communication, International Journal of Man-Machine Studies, 14 (1981) 461-488.

[6] Salton, G., Automatic Information Retrieval, Computer, September (1980) 41-56.

[7] Wente, V., edited by D. Walker, NASA/RECON and User Interface Considerations, in Interactive Bibliographic Search: The User/Computer Interface (AFIPS Press, Montvale, New Jersey, 1971).

CSS 690 / PROJECT REPORT

A MECHANISM FOR MATCHING MULTIPLE PATTERNS IN A TEXT STRING

SUBMITTED TO:

DR. BILL LEIGH

PRESENTED BY:

VIOLET CHEN

THE UNIVERSITY OF SOUTHERN MISSISSIPPI

APRIL, 1985

CONTENTS
****

I. PROBLEM DEFINITION:

This project proposes a mechanism which can match multiple patterns in a run-on text string by modifying the conventional Knuth-Morris-Pratt algorithm which was designed to match only a single pattern in the text string. In addition to a multiple patterns match, this project can handle the replacement of a pattern matched with another pre-determined pattern or the dropping of a chunk of a character string starting from a certain pattern matched and terminating before another pattern match occurs. In the text string, there can be an arbitrary number of patterns appearing. Also, each pattern can appear more than one time. The characters in the string can be any of the 128 ASCII characters in which upper case and lower case letters are considered to be different.

For example, in the following text string:
" apeccccccdunkeyffffffdogggggggggooseeeeeee ",
if we want to substract the fields delimitted by "dunkey" and "dog", meanwhile, replace "dunkey" with "camel" and "dog" with "fox", and with the rest of the character string omitted, the patterns needed to be taken into consideration are: ape, dunkey, dog and goose. On matching the pattern "ape", we start to drop a cunk of character string before we match the pattern "dunkey" and then process the replacement of patterns and substraction of fields before the pattern "goose" is matched. Beginning with the match of "goose", the dropping of characters process takes over.

The new text string results from the match, replacement and dropping process will be the following string:
" camelffffffoxgggggg ".

II. SINGLE PATTERN MATCH MECHANISMS:

A. KNUTH-MORRIS-PRATT ALGORITHM (KMP ALGORITHM)

Compared with a Brute-Force technique, the KMP technique is more efficient, although more complicated to understand for iteratively searching a single pattern in a text string. The algorithms and analysis of it are shown below:

<u>ALGORITHM INITNEXT</u>

Important variables used:

1/. next: Array of the same size as pattern string, except that it stores integer number (as index) in it instead of characters. It is a deterministic finite table produced by matching the pattern string to itself.

2/. p: Pattern String.

3/. i,j: Both serve as index in pattern string array and next table.

4/. m: The length for both pattern length and next table.

Algorithm initnext:

1. set i=1, j=0, next [1]=0
2. while $i \leq m$ do

   check to see fi (j=0) or (p[i]=p[j]), if

   2.1. yes, set i=i+1, j=j+1

check to see if (p[i]=p[j]), if

2.1.1. yes, set next [i] = next [j]

2.1.2. no, set next [i] = j

2.2 no, set j = next [j]

3. exit

ALGORITHM KMPSEARCH

Important variables:

1/. S: text string.

2/. i: index to text string.

3/. p: pattern string.

4/. next: table the same as it is in initnext algorithm.

5/. j: index to both pattern string and next table.

6/. m: length of pattern string.

7/. n: length of text string.

Algorithm kmpsearch:

1. (* initialize index i,j to 1 *) set i=1, j=1

2. (* continue searching until pattern is matched or run out of text string *)

while ($j \leq m$) and ($i \leq n$) do

check to see if (j=0) or (s[i]=p[j]), if

2.1 yes, (* increment both i,j by 1 looking for next match-char *)

2.2 no, (* no matching char detected and j = 0 *)

set j = next [j] (* j index proper position in pattern string according to the value in next [j] *)

3. (pattern matched ?)

   3.1 if $j > m$ then pattern matched, starting with position i-m in text string.

   3.2 if $j \leq m$, pattern is not matched.

ANALYSIS

The Knuth-Morris-Pratt technique, though it is a straightforward algorithm in a sense, is certainly a complicated algorithm to understand. It aims at eliminating the number and length of "false starts" during the searching process.

When a mismatch is detected, the "false start" contains characters we already know by recognizing the value of j which indexes to the character in the pattern string at the point the mismatch occurs. We should be able to take advantage of the information about "false starts" instead of backing up the index over all those false start characters like we do it in the Brute-Force technique.

To avoid the backing up over characters already processed, we need a pre-processing on the pattern to determine which character in the pattern string we should use to start the next matching process. The preprocessing routine, algorithm initnext on page 3, supplies a table called "next" containing the information as to which character in the pattern string we should start for next matching process based upon the position of the character a mismatching takes place. The table "next" is actually a transformation of a deterministic finite table. After running our pattern string on the routine based on algorithm "initnext" we can get a table "next" for the pattern string.

We now use a pattern string to exemplify the execution of INITNEXT. For the pattern string "10000001000000111111", if we step through algorithm "initnext" on it, we will get a table "next" in its transformational form as a deterministic finite table as the one in table 1.A.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	1	1

Next [1] = 0

i j

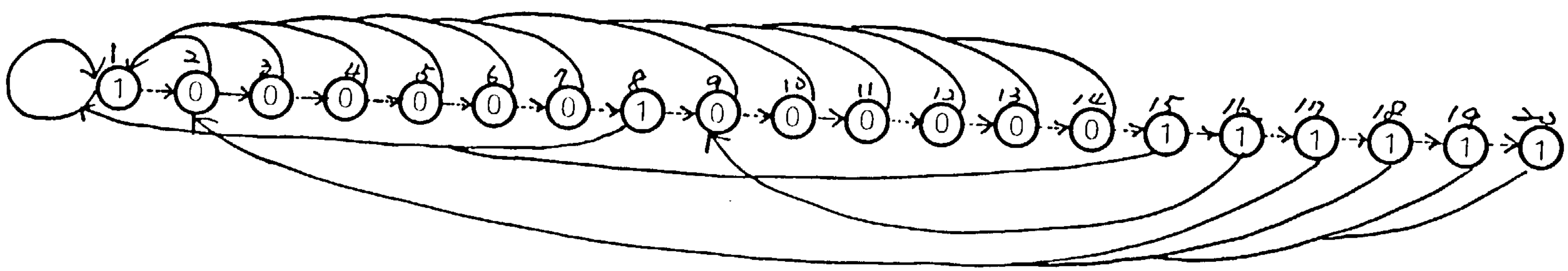
~~1~~ ~~0~~
~~2~~ ~~1~~
Next [2] = 1
~~2~~ ~~0~~
~~3~~ ~~1~~
Next [3] = 1
~~3~~ ~~0~~
~~4~~ ~~1~~
Next [4] = 1
~~4~~ ~~0~~
~~5~~ ~~1~~
Next [5] = 1
~~5~~ ~~0~~
~~6~~ 1
Next [6] = 1
~~6~~ ~~0~~
~~7~~ ~~1~~
Next [7] = 1
~~7~~ ~~0~~
~~8~~ ~~1~~

i j

Next [8] = 0
~~9~~ ~~2~~
Next [9] = 1
9 ~~2~~
~~10~~ ~~3~~
Next [10] = 1
~~11~~ ~~4~~
Next [11] = 1
~~12~~ ~~5~~
Next [12] = 1
~~13~~ ~~6~~
Next [13] = 1
~~14~~ ~~7~~
Next [14] = 1
~~15~~ ~~8~~
Next [15] = 0
~~16~~ ~~9~~
Next [16] = 9

i j

~~16~~ ~~1~~
~~17~~ ~~2~~
Next [17] = 2
~~17~~ ~~1~~
~~18~~ ~~2~~
Next [18] = 2
~~18~~ ~~1~~
~~19~~ 2
Next [19] = 2
~~19~~ ~~1~~
~~20~~ ~~2~~
Next [20] = 2
~~20~~ ~~1~~
~~21~~ ~~2~~
Next [21] = 2



⇢ : transformation on matching character

→ : transformation on mismatching character

Table 1. A "next" table in form of a Deterministic Finite State Graph for a pattern string "100000010000001111111".

Likewise, for another pattern string which is highly self-repetitive is "11111111111111111110", if we step through algorithm "initnext" on this pattern string we can get another "next" table, which we will not illustrate here.

As to the run time for algorithm "initnext", the inner loop statement next [i] - next [j] or statement next [i] = j will be executed definitely M (the length of pattern string) times. Thus, we can say that the worst case running time for algorithm "initnext" is M, in curly oh notation it is notated as O(M).

With the Deterministic Finite state table already established, we can now apply the "kmpsearch algorithm" to accomplish our searching process for the pattern desired to be matched.

At the beginning of the kmpsearch process, both i and j index are initialized to 1., i indexes to the text string and j indexes to the pattern string. As long as i, j index to matching character, both are incremented by 1. If i and j index to mismatching character, we do not really back up i index like we did in Brute-Force searching, but instead, we set j indexing to a proper character in the pattern string according to the index value kept in next [j] indicating the next character in the pattern string we are looking forward to be matched. This is a subtle and tricky point in the Knuth-Morris-Pratt technique. As far as the running time of "kmpsearch" algorithm is concerned, the inner loop statement i=i+1, j=j+1 will be executed N times which is equal to the length of the text string if the pattern is in the last M characters of the text string, which we regard as the worst case.

For example, even for searching a highly self-repetitive pattern "11111110" in an extremely, extraordinarily, highly self-repetitivetext string "11111111111111111111111111111111111111111 111111111111110", the inner loop statement i=i+1 will be executed exactly 56 times (i.e. N times) to accomplish the pattern matching process instead of the 7*48=336 times in the Brute-Force technique, because, i index will never be decremented then incremented, to reprocess the characters already processed several times. In all cases, the running times for Knuth-Morris-Pratt searching technique will be N times plus M times for initializing the "next" table. On the whole, the worst case running time for Knuth-Morris-Pratt algorithm will be M+N. In curly oh notation, it is O(M+N).

For example, to apply the KMP technique for searching a pattern "zigzag" in a string, we need two processes. Firstly, we match "zigzag" to itself by running the pre-processing algorithm INITNEXT to obtain a finite state machine (NEXT) in tabular form (i.e. 0 1 1 1 2 1), indicating which character in the pattern we should go back to retry the pattern match based on the position in the pattern at which mis-match occurred. With the help of the NEXT table, we can run the KMPSEARCH algorithm for searching "zigzag" in the string without backing over any character in the string already searched, until the pattern is searched or run out of string characters which means the pattern is not matched.

B. SNOBOL (A string oriented symbolic language)

SNOBOL which is claimed to be most powerful in string process certainly can be applied to search the above mentioned pattern in a string by only a few simple commands:

```
        TEXT = INPUT
        TEXT @POS ZIGZAG      :S(MATCH) F(UNMATCH)
MATCH OUTPUT = "PATTERN MATCHED AT" POS "POSITION" :DONE
UNMATCH OUTPUT = "PATTERN NOT MATCHED"
DONE END.
```

III. MULTIPLE PATTERNS MATCH MECHANISM

An approach proposed in this project for searching multiple patterns (ex. zigzag, zebra) in a string is a technique which combines the KMP algorithms with a transition table (a finite state machine) technique. In this combination technique, we need to pre-process the patterns twice. For the first time through, we run each pattern on the INITNEXT algorithm to obtain a NEXT table (also a finite state machine) for each pattern. For the second time through, we transform the patterns one character after another, in combination with the mapping from NEXT table associated with pattern, into a big transition table. A big transition table transformed from the above mentioned two patterns; zigzag, zebra will be like the following table:

128 ASCII CHARACTERS

STATES	...	A	B	...	E	...	G	...	I	...	R	...	Z	...
1	0	0	0	0	0	0	0	0	0	0	0	0	2	0
2	1	1	1	1	8	1	1	1	3	1	1	1	1	1
3	1	1	1	1	4	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	5	1
5	2	6	2	2	2	2	2	2	2	2	2	2	2	2
6	1	1	1	1	1	1	7	1	1	1	1	1	1	1
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	1	1	9	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	10	1	1	1
10	1	11	1	1	1	1	1	1	1	1	1	1	1	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0

We always start from state 1 for starting the transformation for each pattern. On match of one character, we transfer to next state already assigned or increment the transition size by 1 as the next state number. On mis-match, we go back to certain state mapping from NEXT table for associated pattern, based on the position of the pattern character mis-match occurs. By using this approach, the size of the transition table will be increased with the increase of the size for each pattern. For each pattern, to map absolute state numbers to the relative state numbers in the NEXT table, an extra table is used to keep track of the pattern transfer patch noted along with the formation of the transition table. In this way, we can help go back to the absolute state number on a character mis-match for each pattern.

IV. LIMITATIONS FOR THIS PROJECT:

A. Both patterns and replace patterns are limited to 80 characters.

B. Grammar table size is limited to 100 (100 patterns).

C. Transition table size is limited to 600 rows and 128 columns.

D. One section of text string is limited to 5000 characters.

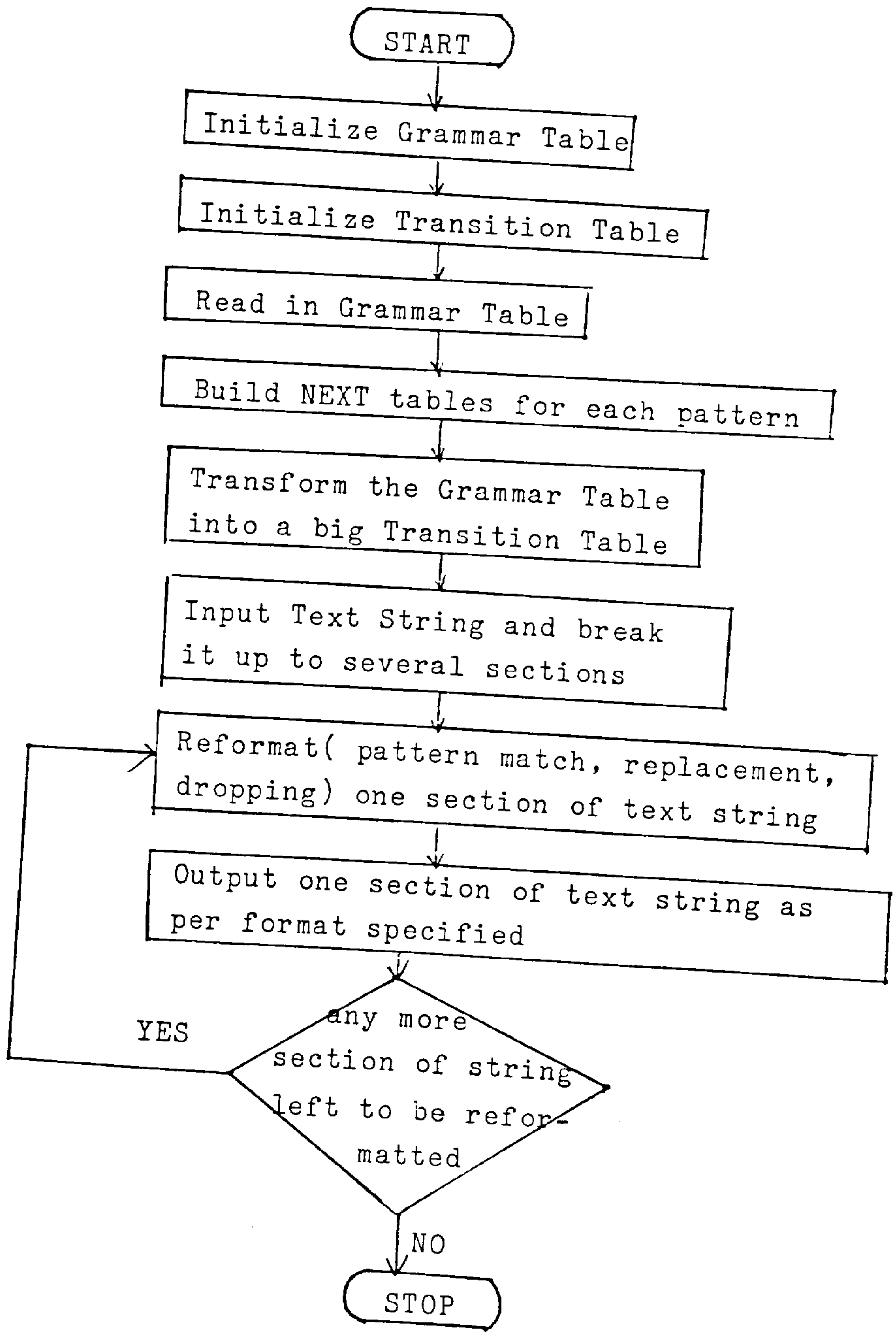
E. A replace pattern "#" is used to specify the beginning of dropping a chunk of character string.

F. Three separate text files are set up for the inputs of patterns, replace patterns and text string respectively.

G. For different set of text string processed, to break each set of text string into several sections, the delimiter used may be different from one set of text string to another. The delimiter specified in main routine needs to be changed for different set of text string with different delimiters.

H. For a specific output format, changes in main routine may be necessary.

V. A MACRO FLOWCHART FOR THIS PROJECT:



VI. **PROGRAM DOCUMENTATION:**

A. INPUT/OUTPUT OF DATA FILES

i. input: 3 text files used.

1. patterns: contains patterns to be searched.

2. repatn: contains assoc. replace pattern for each pattern.

3. teststr: contains text string to be reformatted.

ii. output: to the terminal screen.

B. MODULARIZATION OF THIS PROGRAM:

i. driver routine: main routine.

ii. 6 subroutines:

1. init_grammar_tab

2. init_trans_tab

3. readin_gram_tab

4. build_next_table

5. build_transition_table

6. reformat_strin

For each rubroutine, the pseudo codes highlighting the algorithm will be illustrated in paragraph D.

C. LEGEND FOR IMPORTANT GLOBAL VARIABLES:

1. grammar_tab: array of record, each record contains fields,

1/. patn: pattern to be matched.

2/. len_1: length of the pattern.

3/. next_tab:associated NEXT table foreach pattern.

4/. match_state:state # at which a pattern is matched

5/. replace_patn: associated replace pattern.

6/. len_2: length of the replace pattern.

2. gram_size: grammar table size.

3. trans_tab: transition table, a 2-dimentional array.

4. trans_size: Number of rows representing number of states in the transition table.

5. string_in, string_out: 1-D arrays for input, output section of text string.

6. len_in, len_out: lengths of input, output string.

7. state_patn_map: 1-D array with the same size as the number of states in the transition table for mapping the terminal match state to each pattern.

8. patn_trans_path: keeps track of the absolute state number each pattern transfers through on each character match.

D. PSEUDOCODE FOR EACH SUBROUTINE AND MAIN DRIVER:

Main Driver:

a. call init_grammar_tab subroutine

b. call init_trans_tab subroutine

c. open input mode text string text file

d. call readin_gramm_tab subroutine

e. call build_next_table subroutine

f. call build_transition_table subroutine

g. move spaces to input, output section of strings

h. break up the input text strings into several sections based on certain pre-determined delimiters
   1. call reformat_strin subroutine to reformat one section of string
   2. output one section of string as per format specified
   3. if any more sections of string left to be re-formatted go to step 1

i. exit.

Subroutine init-grammar-tab:

Initialize fields in grammar table, which includes initialize:

1. patterns

2. replace patterns

3. next tables

EXIT.

Subroutine init-trans-tab:

Initialize:

1. transition table elements to -1's

2. state-patn-map to 0's

EXIT.

Subroutine readin-gram-tab:

Read in grammar table:

a: read in patterns:

   1. open input mode pattern text file

   2. initialize gram-size to 0

   3. while not end of patterns file do

3.1 increment gram-size

3.2 initialize length count(i) for each pattern to 0

3.3 while not end of line for each pattern do

3.3.1. increment length count

3.3.2. read in one character to pattern as per position indexed by length count

3.4 store length count i to len-1

3.5 read next pattern line

b. read in replace patterns:

1. open input mode text file of replace patterns

2. initialize gram-size to 0

3. while not end of line for a replace pattern do

3.1 increment gram-size

3.2 initialize length count for a replace pattern

3.3 while not end of line for a replace pattern do

3.3.1. increment length count

3.3.2. read in one character to replace pattern as per position indexed by length count

3.4 store length count i to len-2

3.5 read next replace pattern line

c. EXIT.

Subroutine build-next-table:

Build a NEXT table (finite state machine) for each pattern in the grammar table by matching a pattern to itself. Please see p.2 and pp. 4-8 for the algorithm and analysis for this subroutine.

Subroutine build-transition-table:

Read from grammar table one character after another, one pattern after another in combination with mapping from next tables and patn-trans-path tables to build a big transition table for multiple pattern match.

a. initialize transition table size to 1

b. initialize the 1st row of transition table to 0's

c. for each pattern in the grammar table do

   1. set current state = 1

   2. for each character inthe pattern do

      2.1. keep track of the state at which the character is matched in patn-trans-path table

      2.2. get absolute go back state number by mapping relative state number in the NEXTtable to the state in patn_trans_path

      2.3. next-state contains the value in trans-tab (state, ord(char))

      2.4 check the value of next-state, if next-state equals:

         2.4.1. -1 or absolute goback-state: a new state is created

2.4.1.1. increment trans-size

2.4.1.2. replace the value in trans-tab (state, ord(char)) with trans-size

2.4.1.3. if next-state=-1, we are in a newly created state row, all elements in the row except (state, ord(char)) element are set to go back state

2.4.1.4. set current state to trans-size

2.4.2. other values: set current state = next-state

3. take note of the match state for the pattern just processed

4. set all elements in match state (terminal state) row to 0's

5. in state-patn-map table, mark the terminal state # for the pattern

d. EXIT.

<u>Subroutine</u> <u>reformat-strin</u>:

Input to this subroutine a section of string to be reformatted with string length specified. One character is read at a time and mapped through the transition table for multiple pattern match. An output string is generated along with the matching process with string length updated in the mean time. Once a pattern is matched, the action of either replacing pattern or dropping a chunk of string is taken care of. Current state is reset to 0 to continue another pattern match.

a. initialize variables: initialize

1. knt-in, knt-out, state, last-state to 0's

2. drop boolean value to false

b. while (there is still characters in the section of string to be processed) and (output string length is less than maximum text string length) do

1. process next character in string?

check to see if (state=0) or (state greater than last-state), (note! state=0 indicates the restart to search a pattern, state greater than last state a successful match on a character and ready to match next character in the pattern. State less than or equal to last-state indicates a mis-match on a character and need to go back to a certain character state in the pattern for a successful match.)

1.1. yes: process next character, increment knt-in, check the drop signal, if it is,

1.1.1. on: no write to output string

1.1.2. off: increment knt-out, write input character to output string positioned by knt-out.

1.2. no: remain process on the current input character.

2. set last-state to state

3. state is transferred to the vale in trans-tab (state, ord(char))

4. check if the current state is the match state (terminal state) for certain pattern by checking the value of state-patn-map (state) if it is,

4.1. 0: no pattern is matched

4.2. not 0: a pattern match is detected

4.2.1. find out which pattern is matched

4.2.2. for the pattern matched, check if its associated replace pattern equals "#", if

4.2.2.1. yes: set on dropping signal, drop the matched pattern

4.2.2.2. no: replace the pattern matched with its associated replace pattern in the grammar table. Takes care of due update of output string length count and write.

4.2.3. reset current state to 0 to continue another pattern match

c. EXIT.

E. PASCAL CODES FOR THIS PROGRAM:

The actual pascal coding for this program is as per the computer source listing attached.

VII. BIBLIOGRAPHIC DATABASE TEST DATA EXAMPLES:

A section of input text string can be as the following string sections:

30/5/31

14967 d7407473

'SURELY I CAN IMPROVE ON THAT'(CONCRETE PUMPING EQUIPMENT) CONTRACT. PLANT REV. (GB) VOL.13, NO.4 3-4 APRIL 1974

Treatment: NP
Document Type: 02
Descriptors: CONCRETE; PUMPS
Identifiers: CONCRETE PUMPING EQUIPMENT; VARIABLE DISPLACEMENT PUMP
Class Codes: D8400, D5550

In the above string section, if we want to substract the fields delimited by the pattern "Descriptors:" and "Identifiers:", with the other chunks of string omitted, the patterns chosen to be included in the grammar table can be: "30/5/31", "Descriptors:" "Identifiers:" and "Class Codes:". The pattern "30/5/31" is used toindicate the beginning of dropping a chunk of string characters, the pattern "Descriptors:" indicates both stop dropping and start substraction of character string fields. While, pattern "Class Codes:" indicates both stop substracting and start dropping of chunk of string. For each pattern chosen to be included in the grammar table, there is an associated replace/drop pattern for it indicating what pattern to be replaced or when to start or stop the dropping of string.

The reformatted output string for the above input section of string will look like the following:

```
Key 1: CONCRETE; PUMPS
Key 2: CONCRETE PUMPING EQUIPMENT; VARIABLE DISPLACEMENT
       PUMP
```

A whole set of text string could be like the test data on the computer source listing attached to this page.

VII. BIBLIOGRAPHIC DATABASE TEST DATA EXAMPLES:

A section of input text string can be as the following string sections:

```
30/5/31
  14967   d7407473
  'SURELY I CAN IMPROVE ON THAT'(CONCRETE PUMPING EQUIPMENT)
  CONTRACT. PLANT REV. (GB)  VOL.13, NO.4  3-4  APRIL 1974
  Treatment: NP
  Document Type:  02
  Descriptors: CONCRETE; PUMPS
  Identifiers: CONCRETE PUMPING EQUIPMENT; VARIABLE DISPLACEMENT
               PUMP
  Class Codes: D8400, D5550
```

In the above string section, if we want to substract the fields delimited by the pattern "Descriptors:" and "Identifiers:", with the other chunks of string omitted, the patterns chosen to be included in the grammar table can be: "30/5/31", "Descriptors:' "Identifiers:" and "Class Codes:". The pattern "30/5/31" is used to indicate the beginning of dropping a chunk of string characters, the pattern "Descriptors:" indicates both stop dropping and start substraction of character string fields. While, pattern "Class Codes:" indicates both stop substracting and start dropping ofchunk of string. For each pattern choosen to be included in the grammar table, there is an associated replace/drop pattern for it indicating what pattern to be replaced or when to start or stop the dropping of string.
The reformatted output string for the above input section of string will look like the following:

```
  Key 1: CONCRETE; PUMPS
  Key 2: CONCRETE PUMPING EQUIPMENT;VARIABLE DISPLACEMENT PUMP
```

A whole set of text string could be like the test data on the computer source listing attached to this page.

RETRIEVAL AND PROCESSING

OF

BIBLIOGRAPHIC REFERENCES

USING MICROCOMPUTERS

WITH

SPECIAL ATTENTION

TO

"HYPERTEXT"

by

Steve Howard

May 10, 1985
CSS 690

Graduate Committee:
Dr. Bill Leigh, Chairman
Dr. Cloyd Ezell
Mr. Wayne Walters

"RETRIEVAL AND PROCESSING OF BIBLIOGRAPHIC REFERENCES USING MICROCOMPUTERS WITH SPECIAL ATTENTION TO HYPERTEXT"

Introduction

"Knowledge is of two kinds. We know a subject ourselves, or we know where we can find information upon it" [23, p. 87]. This adage by Samuel Johnson is an appropriate starting point. The ability to seek information is an important component in the larger process of learning and inquiry. Information retrieval and text processing can therefore be simply described as any process which involves the manipulation of information. However, the manner in which this information is "massaged", often becomes rather complex. This report describes those procedures necessary for information retrieval from a bibliographical database using a personal computer. The scope of this report can be described in a threefold manner: (1) a discussion of the history of previous research in text processing and information retrieval, (2) a discussion of the design of a prototype information retrieval system, and (3) a discussion of future trends in this area. In the whole of the discussion, focus is placed on the concept known as "hypertext".

History

During the 1940's, information scientist, Vannevar Bush, designed a system called MEMEX. He observed that no piece of information stood alone: It had a context, such as interconnections of links with a literature on that subject [10, p. 402]. Further, he felt that authors would either follow those links or create new ones. His proposal was to use the technology of his time--microfilm--to store as separate images, each idea or section of a document. Each image was to be made available via a random-access console which he called the MEMEX. This would hold all the documents a person ever read or wrote and would allow the user to peruse "trails" of documents not just by adding or substituting further images but by putting in new trails--new structures [10, pp. 402-403].

Like many grand ideas, Bush's speculations were unimplementable at the time but were resurrected when computer technology began to advance. So during the 1960's, as the enabling computer technology became available, many bibliographic citation and full-text databases covering about any area of knowledge were built and made available for access with computer terminal equipment via telephone lines [11, pp 2-4]. These databases are generally classified into two areas: (1) bibliographic databases and (2) full-text databases. A bibliographic database usually contains citation entries composed of titles, authors, publication information, and abstracts of articles, books, and reports. Full-text databases, on the other hand, contain the complete texts of the documents in addition to the bibliographical information. There is one entry in these databases for each document and usually the databases specialize in one area of knowledge.

Many systems have been constructed in order to provide the user with access to the vast amount of information stored in these databases. In 1965, Theodore Nelson used the idea of Bush's MEMEX to implement what he called HYPERTEXT [10, p. 403]. Nelson's idea behind HYPERTEXT was to store a body of writings as an interconnected whole, with linkages, thus providing instantaneous access to any writings within that body. One early attempt to produce a writer-oriented, computer based system with some resemblance to HYPERTEXT took place in the NLS (on-line system) project. This is now generally referred to by its commercial name, AUGMENT [10, p. 403]. The AUGMENT system is essentially a community of shared files, with facilities for rapid search and linkage. Text documents can be represented as a collection of logical blocks of text, of "statements". Those logical elements are the ideas, be they sentences, paragraphs, diagrams or a row in a table. Superimposed on this basic data structure is a tree structure (analogous to a table of contents), so that users can define hierarchical relationships among the basic building blocks of a document, such as chapter headings, opening and closing paragraphs, and so forth. The MEMEX/AUGMENT ideas are worth remembering for one important reason: They illustrate the possibility of distinguishing between people's conceptions of documents, how a document is presented to a particular user, and how that same document is represented inside a machine.

Another, more contemporary system is THUMB: An Interactive Tool for Accessing and Maintaining Text [16, p. 155]. This system addresses the problem of finding and accessing information stored in the above mentioned databases. Users can access information free from the strictures of linear text, simple indices and page numbers. Retrieval is based on a data structure roughly analogous to a detailed table of contents and a heavily cross-referenced index. The model is created and revised by an expert (such as the author) thoroughly familiar with the represented material. The experts' tasks are made easy by supportive utilities [16].

Still another system of much significance is the ZOG approach to man-machine communications [18, pp. 461-63]. The philosophy behind this style of communication was first developed by the PROMIS (Problem Oriented Medical Information System) Laboratory of the University of Vermont. The basic ideas in ZOG can be described as follows. The user faces a terminal which is displaying a frame. There is text at the top, a list of options below the text, a column of pads at the right side, an area called a workspace below the options, and a horizontal line of pads at the bottom. The user, at his discretion, selects one of the options or pads. Immediately, the frame on the display is replaces by a new one with all the same parts: text, options, vertical column of pads, horizontal line of pads and workspace. Most of the content will be new except for the horizontal pads, which provide a continuously available set of search and help functions.

That is all there is to ZOG as far as external mechanics are concerned. The user traverses a sequence of frames of his own selection, acquiring the information therein and taking the action offered to him. It stands, at this level, simply as a menu selection scheme, distinguished only by its ability to take actions in addition to presenting knowledge.

Like any general purpose interactive programming language, it can serve in any communication capacity: for example, as a command language, data base retrieval system, interrogation system, or question-answering system. Also, like any programming language, what it is good for, as opposed to what it can conceivably be used for, is not determined by it's gross structure, but by the features of operation.

In PROMIS, such a system is used as the sole interface in accomplishing the total set of hospital functions on a ward: keeping patients' medical records, taking patient histories directly from patients, prescribing drugs and treatments, monitoring patient histories directly from patients, prescribing drugs and treatments, monitoring patient progress, checking treatments for side reactions, and retrieving medical knowledge. It is a full range of communication functions with users who range widely in sophistication and in direct skill with the system. The communication interface is only part of the total system that accomplishes all these functions, but it is a central one.

While ZOG was basically a research effort to understand communication between humans and computers, the basic concepts and ideas outlined in ZOG were very influential in the design of other similar systems. One such system is BROWSE: An On-Line Manual and System Without an Acronym. This system consists of a manual database and a retrieval program. The retrieval program provides a basic set of user commands for selecting desired information from menus. To overcome some of the tedious aspects of the basic commands, higher level operations have been defined. Currently only an extremely primitive mechanism exists for producing hard copy output from BROWSE: performing this task "properly" is seen as a hard problem. In spite of this, user acceptance of this system has been quite good [3].

A system that is somewhat more familiar to this author, is SCARS: Southern's Computer Assisted Retrieval Service. This system provides comprehensive access to databases through contracts made with the two largest vendors in the United States: (1) Bibliographic Retrieval Services, and (2) Lockheed's DIALOG system. Currently approximately 250 databases are available, providing access to millions of items. Services are provided in practically all subject disciplines, and with many of the databases it is possible to search a topic across several databases insuring comprehensive coverage of a topic [19].

While all of the systems discussed thus far have been implemented for mainframe computers, only a few similar systems exists for micro or personal computers. Perhaps the most widely known program is FILOS [1, p. 292]. The program FILOS was written in FORTRAN and provides individual users with a flexible and reliable classification system. The various functions of the program may even allow newcomers to create a bibliography of their own, by combining the files of other people. The basic features of the system are shown below [1, p. 287]:

HELP REQUEST

ENTER A NEW REFERENCE

ENTER A NEW REFERENCE WITH REPRINT REQUEST

REPRINT REQUEST FROM A PREVIOUSLY STORED REFERENCE

REFERENCE RESEARCH WITH ONE TO SEVEN KEYWORDS

REFERENCE RESEARCH WITH ONE KEYWORD AND ONE AUTHOR'S NAME

REFERENCE RESEARCH WITH A STRING OF CHARACTERS (UP TO 80)

PRINT RETRIEVED REFERENCE ON THE LINE PRINTER

DISPLAY THE TABLE OF KEYWORDS

INSERT TEXT IN ONE REFERENCE OF DISK B

SUBSTITUTE A STRING OF CHARACTERS IN ONE REFERENCE

RECONSTITUTE DISK A FROM ONE OR MORE EXISTING DISKS B

QUIT THE PROGRAM

Other systems similar to those mentioned above have been developed to aid the user in his search for information. However, those discussed above are a representation of the most widely known and used systems.

Reviewing each of the systems mentioned above in greater detail reveals many similarities to Bush's idea of MEMEX and Nelson's HYPERTEXT. It becomes apparent, indeed, that each is in essence, either an implementation or modification of these ideas.

The AUGUMENT system was one of the first computer based implementations of these ideas. A data structure similar to a tree was used to define the hierarchical relationships described by Bush. THUMB was another implementation of these concepts. This system went one step further however, in that it allowed for a more detailed cross-referencing of the material. ZOG provided the user with a much greater flexibility in his search for information. Still, the basic ideas are simply a modification of those described by Bush and Nelson. The SCARS system relates to these ideas on a much broader scale. In this system, the interconnections are used to link entire documents of similar topics together, rather than similar information within those documents. Finally, FILOS is a system which simply implements these concepts on a smaller scale--for micro or personnel computers.

While each system varies widely in such areas as the data structures used, features of the system, and in the manner and methods of conducting searches, each is basically a manifestation of those ideas outlined by Bush's MEMEX and Nelson's HYPERTEXT. In other words, each system appears to be founded on the basic concept that no piece of information stands alone--all literature on a subject contains links or interconnections to other literature on that subject [10, pp.402-403].

A Prototype for Personal Computers

The classification of bibliographic references is an important but often time-consuming task for most persons. Very often, this work is not rewarding because the classification system does not allow for fast and reliable retrieval of references. In fact, most manual systems become extremely cumbersome when more than a few references have to be retrieved. This is not necessarily due to improper information storage, but rather to inefficient research and searching procedures. These problems can be solved by computers, which have fast access to the data and are designed for string manipulation. However, computers with programs for storage and retrieval of bibliographic references are generally not accessible to everybody [1, pp. 285-86].

Recent advances in the technology of personal computers, combined with their general availability and widespread use, make it an ideal tool for use in information storage and retrieval. Therefore, it appears useful to implement a program for the storage and retrieval of information on a micro-computer.

One of the major benefits to be gained in such a system is in the case where no adequate library is available or easily accessible. In this instance, the desired information can be easily stored on some type of magnetic storage media, such as a floppy diskette, and accessed later from comfortable surrounds and at a time convenient to the user. The other obvious benefit is in the facilitation of storage and retrieval of bibliographies. Such a system would greatly reduce the cumbersome tasks of searching through the indexes of books looking for one particular area of information. Numerous other benefits are to be gained from such a system--many of these will be discussed elsewhere in this paper.

The software for this prototype was developed under the PC-DOS operating system, using an IBM Personal computer. The program requires 128K of random access memory, at least one double sided double density disk drive, and is written in BASIC. The primary goal of this project was to create a program which is easy to use, yet powerful enough to provide the user with the desired information. No attempts were made to produce the most finely tuned and efficient system. Rather, an attempt was made to implement some of the ideas described[1] earlier in this paper--

(1)A simple method for speeding up the term detection phase of retrieval from a full-text document database is presented in "A Method for Speeding Up Text Retrieval," by Per-Ake Larson. This article makes mention of the use of a surrogate database, in which a document is represented as a sequence of hash signatures. These methods greatly increased the speed of retrieving articles from the database.

especially those ideas related to Nelson's HYPERTEXT.

The following is a list of the basic features of this program:

INSERTING NEW CITATIONS INTO THE DATABASE

SEARCHING THE DATABASE BY TOPIC

SEARCHING THE DATABASE BY TITLE

SEARCHING THE DATABASE BY AUTHOR

SEARCHING THE DATABASE BY PUBLISHER

SEARCHING THE DATABASE BY PUBLICATION DATE
- LIST ALL CITATIONS PUBLISHED BEFORE A PARTICULAR DATE
- LIST ALL CITATIONS PUBLISHED AFTER A PARTICULAR DATE
- LIST ALL CITATIONS PUBLISHED ON A PARTICULAR DATE

DISPLAY ALL KEYWORDS

DISPLAY ONE KEYWORD

DISPLAY CITATION(S) BY KEYWORD

SEARCH FOR MULTIPLE WORD PHRASES

SEARCH FOR MULTIPLE KEYWORDS

INSERTING CITATIONS INTO THE DATABASE. Citations were designed to be entered into the database in one of two ways. The first and most cumbersome way is to enter the information via the keyboard under program guidance. The second manner is to retrieve information from a large mainframe database and "download" the information to a personal computer--the task at this point being to determine which citations are most relevant to the user's particular needs.

(2)

Once all the citations have been entered, a new keyword file is created for the entire database. Unlike most presently available systems which allow the user to specify up to ten keywords or strings for which the file will be searched, this keyword file contains every word of the citation (words with insignificant meaning such as "a", "and", "the", etc. are omitted from the file). Although this method of implementation requires considerably more storage space, it does not limit the user to searches involving only those keywords found in a limited keyword table. This method also eliminates the need of the user having to select a limited number of keywords which will be used to classify a particular citation.

RETRIEVAL OF CITATIONS USING BIBLIOGRAPHICAL INFORMATION.

Figure 1 shows an example of a terminal display screen. Once the information displayed in the diagram has been entered, the program will allow the user to retrieve references from the database in the following ways.

---

(2) The only requirement necessary to process citations in this prototype, which were downloaded from a mainframe, is that they be in ASCII format. Once they have been saved on disk, the necessary modifications--such as placing the bibliographical information in the proper format, and placing quotations around the text of the citation--can either be performed by hand or with the aid of a separate program. A rather simple program could be written to accomplish most, if not all, of these operations.

The first method of retrieval is one of accessing the abstract by topic. In this method, the user is prompted to input the topic of the articles he desires to locate. If any articles contained in the database match the requested topic, the entire citation will be displayed on the CRT. This process continues until there are no more citations which match the requested topic. Searches can be conducted in a similar manner for the following categories: author's name, title of the citation, and the publisher of the citation. A search by date is also provided. This search is somewhat different from those mentioned above. A search by date can be "broken down" to include the following: (1) a search for all articles published on or before a particular date, (2) a search for all articles published on or after a particular date, and (3) a search for all articles published on a particular date. Once found, these citations will also be displayed on the CRT. Program generated menus and prompts guide the user through these various searches.

KEYWORD SEARCHES. Searching the database can be performed based on a keyword search or a string search. The keyword search uses the keyword file to access all records related to the keywords requested by the user. The string search scans the file sequentially, looking for all records which contain the requested strings. Which method is most appropriate will depend upon the needs of the user.

A program generated menu will be displayed at this point, in order to provide the user with a set of operations necessary to

FIGURE 1: Example of Terminal Display.

INDEX NUMBER	:	A-1
AUTHOR	:	Weyer, Stephen A.
TITLE	:	THE DESIGN OF A DYNAMIC BOOK FOR INFORMATION SEARCH
PUBLISHER	:	Academic Press
DATE PUBLISHED	:	1982
TOPIC	:	Information Retrieval
ABSTRACT	:	Information-seeking skills are central to many learning tasks, from finding simple facts and comparing events in an encyclopedia, to pursuing complex research questions. This article describes a simple dynamic book, based on a world history textbook.

structure the information environment in a useful manner. The first menu selection, simply provides a list of all keywords found in the database and their relative citations. Figure 2 provides an example of the screen display.

The second selection is similar to the first. However, it provides the user with the capability of viewing the related information about one specific keyword, rather than the entire list. A display screen similar to Figure 2 will list this information. If the requested keyword was not found in the database, an appropriate error message will be displayed and the user given the option of entering a new keyword or returning to the keyword menu.

The remaining three selections provide the user with perhaps the most powerful tools found in this program. It is in this section of the program that Nelson's ideas of HYPERTEXT (storing a body of writings as an interconnected whole, with linkages, and... providing instantaneous access to any writings within that body [10, p. 403]) are implemented.

These interconnections within a body of writings can be pictured in much the same manner as planning a trip and navigating a course toward some destination. The original destination may be too vague and in need of further specification, or if it is very distant, the traveler may choose to break the trip into a series of smaller, connected trips.

FIGURE 2. Keyword listing.

KEYWORD LIST

KEYWORD	ABS.	ABS.	ABS.	ABS.	ABS.	ABS.	ABS.	ABS.	ABS.
ARRAY	A-1	A-2	A-3						
ART	A-4								
ARTICLE	A-1	A-2							
GOVERNMENT	A-2	A-4							
INFORMATION	A-1	A-4							
PROGRAM	A-2								
TERMINAL	A-1	A-2	A-3	A-4					

PRESS ANY KEY TO VIEW MORE KEYWORDS

During the trip, the traveler may decide to change destinations or vehicles, explore or enjoy the surrounding terrain, travel in circles over old territory, reach impasses and backtrack, or finally arrive at some destination [23, p. 88].

To aid in search (as do maps for navigation), information should be structured in such a way as to reflect the underlying natural order of the article. Pieces of information are related to each other by their physical proximity in a paragraph, on a page or on neighboring pages. A subject index provides access to parts of the book in some other order. A good teacher, a set of questions or the authors can help provide connections and cross references to seemingly distinct sections and ideas; footnotes (and parenthetical remarks) refer to details of minor interest, named references to figures and chapters lead to other pages, and bibliographic citations point to other books or articles [23, p. 88].

Selection three of the keyword menu, provides the user with the ability to display all the citations which are found to contain a requested keyword. The fourth selection provides a similar concept. It however, allows the user to search for multiple word phrases which might occur in the citations, rather than simply one keyword. Phrases up to ten words in length can be specified.

In the final menu selection, the user is provided with the most versatility. After selecting an initial keyword, the citation will be displayed on the CRT with every occurrence of the keyword highlighted in reverse video. At this point the user has the following options: (1) he may press the enter key and continue viewing any remaining citations for the requested keyword, or (2) he may use the space bar to position the cursor over any new word in the citation. Once the desired new keyword has been selected, the escape key can be pressed, thus allowing a search to be conducted for the new keyword starting at the beginning of the database. This process can be endlessly repeated, until the necessary information has been acquired or is proven to not exists, thus providing the user with a powerful, yet easy to use method of navigating through the database. A hypothetical session showing the features of the HYPERTEXT implementation can be found in appendix B.

Although the program was originally designed for an IBM-PC, the basic concepts can essentially be implemented on any microcomputer. The program was not intended to compete with the various bibliography programs running on large computers. It should rather be viewed as a convenient way of storing and retrieving bibliographic references on a moderate scale.

"HYPERTEXT" with PF474

A product for the IBM-PC which could remarkably enhance the performance of the above HYPERTEXT implementation is PF474. It is available from Proximity Technology Incorporated, Fort Lauderdale, Florida. This product helps solve the problem of finding approximately matching strings. A better understanding of the challenges involved in finding approximately matching strings can be found in the introduction of the PF474 Product Data Book.

Most data from human input, or physical processes comes in an inexact form, which must then be interpreted. The processing of information based on inexact, incomplete or inaccurate data has been a difficult problem for computers to cope with. The fundamental component of many tasks from finding a name in a database to optical or speech recognition is that the computer must be able to perform pattern matching strings. That is, it must be able to recognize and retrieve strings which are similar or approximately equal to a query string.

In the past, solutions to this kind of problem have employed special purpose algorithms which are limited in their capabilities. Such algorithms normally require that the strings being compared vary by no more than an arbitrary number of deletions, insertions and substitutions of letters. These software implementations are constrained by the requirement to rapidly locate a similar string within a database of thousands, perhaps tens of thousands of entries.

Generally, such a system can be conceptually broken down into two parts: first, a method of limiting the search space to a small number of strings and second, a string matching function which identifies the most similar records in the restricted search space. The search space is restricted because the string matching algorithms have only been able to make up to a few thousand comparisons per second. Unfortunately, restricting the search space reduces the effectiveness of the system by decreasing the thoroughness of the search. Hence, the system may work fine with differences of up to a certain degree, but beyond that, will totally fail.

Take, for example, the problem of redundant records in a database. A most common example is the duplication of entries in a mailing list because some element of the name or address is not entered consistently (how many times have we received a duplicate mailing or solicitation to apply for a credit card we already have?). Consider the following addresses:

John Q. Public 123 Northwest Main Saint Duplicate, USA	JQ Public 123 N.W. Main ST. Dupilcate, USA

It is apparent to a human that these two addresses are for the same person due to our innate ability to recognize patterns (in fact, you do this so automatically that you probably didn't notice the transposition of 'i' and 'l' in the last line of the address). However, the limited software solutions described above would be confounded by the number of insertions/deletions and other discrepancies in the two addresses. Computers have been unable to duplicate this elusive element of human intelligence[24,p.1].

A description of PF474 follows:

The PF474 microcircuit is a totally new device which implements the revolutionary String Proximity Computer and Ranker on a single VLSI silicon chip using high-speed NMOS technology. The String Proximity Computer compares two symbol strings to arrive at a numeric rating of their similarity: a 32-bit binary fraction ranging between zero and one. It is useful to think of the PF474 as a scoring device that produces high scores for very similar strings and low scores for highly dissimilar strings. In contrast to older methods, the Proximity Function offers a measure of similarity between strings that is both fast and meaningful.

The Proximity Function, as implemented in the PF474, is very flexible and highly adapts to specific problems. This is important because an optimal comparison method for one type of input like alphanumeric text will not be as effective in comparing data such as encoded audio or video signals.

In most applications, one string is chosen as a search string or query. This query string is compared with each string in a large database by the PF474, which computes a Proximity Value for each of the database strings. The sixteen highest Proximity Values (corresponding to the best matches) are stored in a ranked list in the PF474 for reference at the end of the search.

The Proximity Function is unique among string matching algorithms in that it can be computed quickly in hardware. Furthermore, the comparison time is linear with the number of characters in the strings. Thus, for strings of length 8, the comparison rate is 59,500 comparisons per second, and for strings of length 127, the comparison rate is 4,600 per second. (These speed figures assume a system providing strings to a 3.6 MHz PF474 at its full input rate.) Thus many search problems may be solved by exhaustive methods within an acceptable amount of time. This is quite remarkable since other similar functions generally require a computation time proportional to the square of the length of the strings[24, pp 1-2].

Some of the features and capabilities of PF474 include:

- Computes 32-bit Proximity Values
- Parameter tables, stored in externally accessible RAM, allow tailoring of the Proximity Function
- Maintains internal 16-element ranked list of best matches
- Permits location of 15 additional next best matches
- Modern programming architecture allows the device to be accessed as normal memory
- High-speed DMA facility permits rapid loading from external memory: up to 2 million bytes per second
- Smart DMA permits optional editing of the input data to offload certain useful pre processing tasks from the host processor
- Fits naturally and simply into microprocessor based systems[24, pp.2-3]

Although PF474 has a vast potential of applications, it can easily be seen how such a product could greatly enhance the overall performance of the prototype described earlier in this paper. PF474 could be used in comparing the keywords entered by the user to those contained in the database of citations. Instead of providing the user with only those keywords which exactly match the keyword entered, the user could be provided with an option to view those words which had the highest proximity value to the requested keyword. Beside the obvious advantage of vastly reduced search time, this would also provide the user with one more tool in his search for information.

Future Work and Conclusions

There remain some research questions. One, still, is the nature of searching. How do people search? What makes them decide to take certain action? What effects does the particular choice of commands or even the manner in which a machine converses with a user have on the user's performance? Even with humans in this role, when is it successful and when not? What can be done by intermediary or user to enhance the interaction between the two? [15, p. 59]

These questions as well as others will need to be researched. It appears that this is certainly an expanding area for the application of microcomputers, and one that will continue to provide many challenges in the development of such systems.

A P P E N D I X   A

BASIC source code

listing of the program

APPENDIX B

Hypothetical Session Showing Features of "Hypertext" Implementation

<<< KEYWORD MENU >>>

1. Display All Keywords
2. Display One Keyword
3. Display Abstract(s) by Keyword
4. Search for Multiple Word Phrase
5. Search for Multiple Keywords
6. Return to Main Menu

Enter Selection: 5

The user should enter selection 5 in order to use the HYPERTEXT implementation.

After keying in the appropriate selection and pressing the enter key, the following screen will be displayed:

ENTER KEYWORD FOR WHICH YOU ARE LOOKING?

At this point the user should enter the desired keyword (topic). A search is initiated beginning with the first entry in the database. If the keyword was not found in the database, the following message will be displayed on the CRT.

KEYWORD NOT FOUND IN ANY ABSTRACT
WOULD YOU LIKE TO SEARCH FOR ANOTHER (Y/N)?

Entering YES at this point will cause the message found at the top of this page to be displayed, NO will cause control to be returned to the menu on the previous page.

If the keyword was found in the database, each citation containing the keyword, will be displayed on the CRT with every occurrence of the keyword highlighted. At this point the user has the following options: (1) he may press the enter key and continue viewing any remaining citations for the requested keyword, or (2) he may use the space bar to position the cursor over any new word in the citation. Once the desired new keyword has been selected, the escape key can be pressed, thus allowing a search to be conducted for the new keyword starting at the beginning of the database. This process can be endlessly repeated, until the necessary information has been acquired or is proven to not exists.

Suppose a user has downloaded citations on the topic of dairy products. At this point he wishes to go through the database using the hypertext feature looking for specific information related to cheese. The keyword cheese would be entered and the search would begin. Only those citations which relate specifically to the keyword cheese would be displayed. At some point during the search the user realizes that there are a number of varieties of cheese discussed in the citations. He may then move the cursor to select a specific type of cheese--Swiss, for example--as the new keyword. A new search would then begin from the beginning of the database providing the user with only specific information on Swiss cheese rather than cheese in general. This procedure could then be repeated for other varieties of cheese, thus providing the user with the desired information centering around a very specific topic. The main advantage to be realized by such a system, is the flexibility provided the user in perusing these different "trails" of information which are threaded throughout the database. A sample display can be seen on the following page.

Example of Terminal Display.

INDEX NUMBER	:	C-4
AUTHOR	:	Mickey Mouse
TITLE	:	The Nutritional Value of Cheese
PUBLISHER	:	M. Mouse Press
DATE PUBLISHED	:	1985
TOPIC	:	Dairy Products
ABSTRACT	:	Information may be contained here about Swiss cheese and it's nutritional value. Also information about other types of cheese would be contained in the remainder of this citation.

PRESS RETURN TO VIEW MORE ABSTRACTS
PRESS SPACE BAR TO SELECT NEW KEYWORD,
ESC TO BEGIN SEARCH

REFERENCES

[1] Bertram, D., and Bader, C., "Storage and Retrieval of Bibliographic References Using a Microprocessor System", **INTERNATIONAL JOURNAL OF BIO-MEDICAL COMPUTING**, Vol. 11, 1980.

[2] Bonczek, Robert H., C. W. Holsapple and A.B. Whinston, **FOUNDATIONS OF DECISION SUPPORT SYSTEMS**, Academic Press, Inc., Harcourt Brace Jovanovich Publishers, Orlando, 1981.

[3] Bramwell, Bob, "BROWSE: An On-line Manual and System Without a Acronym", **ASTERISK** (The official publication of the special interest group for systems documentation), December 1983, Volume 9, Number 4.

[4] Cote, Alfred J., **THE SEARCH FOR THE ROBOTS**, Basic Books, Inc., New York, 1967.

[5] Hayes-Roth , Frederick, "The Role of Partial and Best Matches in Knowledge Systems", Academic Press, Inc., 1978.

[6] King, Richard P., Henry K.Korth, and Barry E. Willner, "Design of a Document Filing and Retrieval System", **DATA BASE**, Winter 1984.

[7] Lancaster, F., **INFORMATION RETRIEVAL SYSTEMS**, Wiley-Interscience, New York, 1979.

[8] Larson, Per-Ake, "A Method For Speeding Up Text Retrieval", **DATA BASE**, 1984.

[9] Lefkovitz, David, "PRIMATE: A Microcomputer System for Host Mediation and Private File Text Search", **Proceeding of ASIS Annual Meeting**, Volume 19, 1982.

[10] Lefrere, P., "Text Processing", in **ARTIFICIAL INTELLIGENCE**, edited by T. O'Shea and M. Eisenstadt, Harper and Row, New York, 1984.

[11] Leigh, William and Michael Berry, "Computer Applications in Information Retrieval and Writing for Technology Transfer", (unpublished article to appear as a chapter in **COMPUTER APPLICATIONS FOR THIRD-WORLD COUNTRIES** to be published by MacMillan in 1985).

[12] MacDonald, N. Frase, L., and Keenan, S., "Writer's Assessment", Bell Laboratories, Piscateway, New Jersey, 1980.

[13] Marcus, Richard S., Peter Kugel, and Alan R. Benenfeld, "Catalog Information and Text as Indicators of Relevance", **JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE**, John Wiley and Sons, Inc., 1978.

[14] Maron, M., and Kuhns, J., "On Relevance: Probabilistic Indexing and Information Retrieval", **JOURNAL OF ASSOCIATION FOR COMPUTING MACHINERY**, Vol. 7, 1960.

[15] Meadow, Charles T., "The Computer as a Search Intermediary", **ONLINE**, July 1979.

[16] Price, L., "Thumb: An Interactive Tool for Accessing and Maintaining Text", **IEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS**, Vol. SMC-12, No. 2, March/April 1982.

[17] Raphael, Bertram, "Natural Language" in **THE THINKING COMPUTER - MIND INSIDE MATTER**, W. H. Freeman and Company, San Francisco, 1979.

[18] Robertson, G., McCracken, D., and Newell, A., "The ZOG Approach to Man-Machine Communication", **INTERNATIONAL JOURNAL OF MAN-MACHINE STUDIES**, Vol. 14, 1981.

[19] "SCARS", (unpublished brochure outlining the SCARS system).

[20] Sarin, Sunil K., and Irene Greif, "Software for Interactive On-Line Conferences", **ACM**, 1984.

[21] Spencer, Donald D., **COMPUTERS IN SOCIETY**, Hayden Book Company, Inc., Rochelle Park, New Jersey, 1974.

[22] Wente, V., "NASA/RESON and User Interface Considerations", in **INTERACTIVE BIBLIOGRAPHIC SEARCH: THE USER/COMPUTER INTERFACE**, edited by D. Walker, AFIPS Press, 1971, Montvale, New Jersey, pp. 95-104.

[23] Weyer, Stephen A., "The Design of a Dynamic Book for Information Search", Academic Press Inc. (london) Limited, 1982.

[24] **PF474 PRODUCT DATA BOOK**, Proximity Technology, Inc., 1984, Fort Lauderdale, Florida.

ECONOMIC DEVELOPMENT AND TECHNOLOGY TRANSFER: MISSISSIPPI'S APPROACH

G. David Huffman, University of Southern Mississippi
David Murphree, Mississippi Institute for Technology Development

VITA

Dr. G. David Huffman is Dean, College of Science and Technology and a Professor of Computer Science at the University of Southern Mississippi. Dr. Huffman was previously Chairman and Professor of Computer Science at Purdue University, Indianapolis, Chief Scientist at the Indianapolis Center for Advanced Research, Distinguished Visiting Scientist at the Air Force Academy and Chief, Mechanics Research, the Allison Division of the General Motors Corporation.

Dr. Huffman received his B. Engr. S. (cum laude) from Marshall University and his M.S. and Ph.D. (Mechanical Engineering) from the Ohio State University. Dr. Huffman carried out postdoctoral studies at the Imperial College of Science and Technology.

Dr. David Murphree is the President and Chief Executive Officer of the Mississippi Institute for Technology Development. Dr. Murphree was previously Director of the Mississippi MHD Energy Center and a Professor of Aerospace Engineering at Mississippi State University. Dr. Murphree received his B.S. (Aeronautical Engineering) from Mississippi

State University and his M.S. and Ph.D. (Physics) from the University of Wisconsin.

ABSTRACT

The Mississippi Institute for Technology Development (ITD) is a new venture in technology-based economic development. ITD is a partnership between the federal government, the state of Mississippi, and the comprehensive universities of Mississippi. ITD incorporates the significant characteristics of successful research and development centers, i.e., a strong university relationship, a sound research and financial base, available research personnel, an applied research focus on activities having maximum commercial potential and strong governance. ITD has now established three technology centers: the Center for Advanced Building Systems, the Center for Applied Diagnostics and the Center for Polymers. In addition to the above Centers, ITD is initiating a technology transfer program which includes support of local industries through contract research and development, commercialization of new products and technology applications and assistance studies. ITD has the total support of state leaders and Mississippi looks forward to a revitalized, technology-based economy.

INTRODUCTION

The Mississippi Institute for Technology Development (ITD) is a private, non-profit corporation with the stated objective of conducting and transferring scientific research into usable technology for commercial application in order to improve the economic development of Mississippi and the nation, Murphree (1983) and Mississippi Implementation of Public Law 96-480 (1980). The formation of ITD by Mississippi leaders was premised on: an inability by Mississippi to be domestically and internationally competitive as a low-technology, low-wage state, technology-based economic development was essential to the state's future, the research and development (R&D) gap between Mississippi and other parts of the country was large and was growing, an expanded R&D base in the state was essential for technology-based economic development, most university-based research activities lacked a commercialization thrust, capable research centers existed in Mississippi but lacked a critical mass, the state possessed the support services and economic development climate requisite for enhanced technology commercialization and that substantial external funding would be required to match state and private funds.

At this point in time, ITD has received both federal, state and private funding. An organization structure is in place, technology divisions have been formed and personnel are being hired. These topics along with future prospects are discussed in the following sections.

CHARACTERISTICS OF RESEARCH AND DEVELOPMENT CENTERS

Since ITD represents another--albeit unique--approach to technology-driven economic development, it is useful to review the general characteristics of existing and/or established R&D centers;* A.D. Little (1984). Historically, most organizations have developed as an outgrowth of university research programs or as a result of significant federal or state funding, e.g., SRI is an example of the former with the Indiana Corporation for Science and Technology illustrating the latter. The bulk of existing R&D centers are oriented toward increasing the local or national technology base with a secondary mission of economic development. The economic development and technology transfer component has been increased, however, in the newer organizations, eg., ITD (Mississippi), Advanced Technology Development Center (Georgia), New York Science and Technology Foundation (New York), Ohio Technology Transfer Organization (Ohio), Ben Franklin Partnership Fund (Pennsylvania), etc.

While legal status and organization structure vary among the R&D centers, most are non-profit corporations with a strong Board of Directors. Board members are drawn from the

*A generic term denoting university, state or independent organizations with a mission of research, development and technology transfer leading to regional or national economic development.

private, university and government sectors. As noted above, most organizations evolved from a university setting and continue to maintain a strong university relationship and/or affiliation. It is important to note, however, that the objectives of the university and the R&D Center may not coincide on some issues and it is important that the R&D Center be able to establish its own agenda. From an administrative point-of-view, the ability of the R&D Center to implement its own personnel policies, purchasing functions, etc. is an important consideration.

Most organizations carry-out both basic and applied research programs. The successful organizations tend to emphasize the latter and have a clear recognition of the need to focus their activities on specific research and/or development fields. Funding levels vary from around $1 million/year for industry/government research centers to levels of approximately $50 million/year for a few government sponsored research facilities.

Assessing the economic development resulting from an R&D center is a difficult proposition at best. Many organizations are relatively new and economic development is a long-term process. Furthermore, corequisite conditions such as the availability of venture capital, personnel with entrepreneurial abilities, critical mass of engineers and technicians, etc. must also be satisfied for economic development to occur. Despite the uncertainties, most organizations believe that an R&D center improves the attractiveness of the local area to outside technology-based companies and strengthens the competitiveness of existing organizations.

According to A.D. Little (1984), successful R&D centers demonstrate the following characteristics: a strong university relationship with a formal or working agreement, a sound research base at initiation encompassing financial commitments, grants and contracts in hand and a successful research capability, available research labor including research and teaching assistants, an applied research focus on activities having commercial potential, strong governance with a board which is both active and respected, and industrial support including both funding and decision-making.

THE MISSISSIPPI INSTITUTE FOR TECHNOLOGY DEVELOPMENT

As noted above, ITD is a not-for-profit corporation with the mission of fostering economic development through research, development and technology transfer. ITD works closely with both Mississippi and national organizations. In particular "ITD will interface the needs of the Federal Government, State Government, foundations and industries with the R&D expertise of the universities and together with technology transfer from national laboratories, promote commercial development of new products in Mississippi and the nation".

ITD incorporates those factors believed to be essential for a successful R&D center. The Institute has now established working relationships with Mississippi State University, the University of Southern Mississippi, the University of Mississippi and Jackson State University. These universities provide a sound research base and a pool of available skilled researchers and support personnel. ITD has also established liaison with federal laboratories in Mississippi and is working closely with NASA's National Space Technology Laboratories in the area of remote sensing. Federal state and private funding amounting to $38 million is now available providing a strong initial financial base. Furthermore, ITD working with the Mississippi R&D Center and university faculty is implementing a broad-based economic development plan.

In an attempt to focus activities consistent with the funds available and the greatest potential for commercial development, ITD initial technical activities will be limited to three technology centers: the Center for Advanced Building Systems, the Center for Applied Diagnostics and the Center for Polymers. The Center for Advanced Building Systems will include work in the areas of computer-aided-design and manufacture (CAD/CAM) of housing, design and repair of infrastructure--roads, highways, bridges, etc., building materials and living systems. The Center for Applied Diagnostics is based around remote sensing and its application to a wide range of diagnostic problems in combustion/propulsion, atmospheric sciences, ocean sciences, geo-sciences, medicine and animal sciences. The Center for Polymers will focus initially on coatings, military materials, plastics/composites and specialty applications. This Center will also provide support to the other Centers.

In addition to the research and development activities described above, ITD will also perform a technology transfer function. This will consist of: support of local industries through contract research, commercialization of new products and/or processes and technology application and assistance studies. Morevover, ITD will also provide an inventions managemement function and--in cooperation with the Mississippi R&D Center--a business development function.

To summarize, the Mississippi Institute for Technology Development has been formulated to maximize its chances of success, e.g., incorporating the most significant characteristics of existing successful organizations. Federal, state and private funding has been obtained, technology centers formed and initial staffing is underway. Mississippi views this initiative as mandatory and looks forward to a revitalized technology-based, economic development program.

REFERENCES

[1] Murphree, D. "Institute for Technology DevelopmentState of Mississippi", July, 1983.

[2] "Mississippi Implementation of Public Law 96480", October 21, 1980.

[3] A.D. Little, Inc. "Feasibility Study of the Mississippi Institute for Technology Development", March 30, 1984.

DEVELOPMENT OF A TECHNOLOGY TRANSFER WORKSTATION

William Leigh
G. David Huffman
Noemi Paz
Dennis Vital

University of Southern Mississippi
Computer Science Department
Hattiesburg, Mississippi 39406
(601) 266-4949

DEVELOPMENT OF A TECHNOLOGY TRANSFER WORKSTATION

ABSTRACT

For the past eighteen months, we have been developing a "Technology Transfer Workstation." This has resulted in a set of personal computer hardware and software to support the endeavors of bibliographic database searchers, which is a major activity in research and technology transfer. The specific brand of database searching of concern to us is that of NASA and its sub-contractors.

We proceeded by: 1) specifying the needs; 2) surveying and selecting commercially and otherwise available products which might be pieces of the solution; and 3) building the pieces (software only) of the solution which we could not find commercially.

Progress to date includes the development of components of a workstation for a VAX minicomputer and the installation of a prototype workstation on the IBM-PC. The IBM-PC implementation offers the most utility for NASA's purposes. For the coming year, we propose to tune the performance of this PC version and to make it available to selected sites for evaluation.

DEVELOPMENT OF A TECHNOLOGY TRANSFER WORKSTATION

W. Leigh, G. D. Huffman, N. Paz, D. Vital

This paper reports the results of work carried out by us as part of a project to enhance the productivity of NASA's technology transfer arms, the Industry Application Centers (IAC) and the State Technology Assistance Centers (STAC). Both of these types of organizations exist to make available to the public the results of NASA technology developments. Concomitant to that major goal, the IAC's and STAC's often transfer the results of technology development by groups other than NASA.

In all of these cases the usual mode of operation is the bibliographic database search, often augmented by the direct contacting of experts and the writing of engineering type reports. Recent advances in the technology of personal computers and the continuing decline in their costs prompted the investigation of the employment of personal computers in the facilitation of this work, specifically to assist in the searching of the databases, in the sorting of the resulting citations, and in the writing of the report.

1. BIBLIOGRAPHIC DATABASE SEARCHING AND TECHNOLOGY TRANSFER

We regard the NASA technology transfer process as an information system. A primary output is known as the "industry applications study."

The first step in the development of an industry applications study is the formulation of the problem statement. Once the problem is formulated, a series of bibliographic databases are selected for searching. These databases may be either commercial, e.g., DIALOG, or public domain, e.g., NASA RECON. Using database thesauri, the search strategy is formulated and a series of keywords chosen. An on-line search is conducted and a number of abstracts retrieved. These are then reviewed for relevance and a number of documents ordered. The documents can then be analyzed and, if warranted, government and/or industrial contacts instituted. Information from experts in conjunction with the assessment of the published information is used to formulate the final report.

Another primary output is the "current awareness search." The steps carried out in a current awareness search are similar, but normally terminated after the on-line search, i.e., the final report consists of the retrieved abstracts. Presumably, the recipient of the abstracts carries out the abstract review and analysis procedures and contacts the relevant experts.

The labor intensive steps in the above process are associated with the review of the abstracts for relevance, the analysis of the relevant publications and the preparation of the final report. Analysis and report writing labor can be reduced through the employment of computer information systems tools and techniques.

2. SOFTWARE FRAMEWORK FOR TECHNOLOGY TRANSFER

Our goal is to develop a computer-based "technology transfer workstation" of productivity tools for technology transfer as it at NASA Industrial Applications Centers (IAC) and State Technology Assistance Center (STAC). We are seeking and evaluating available packages for this purpose as well as developing programs to meet the needs for which no suitable packages can be found. Our main goal is to identify how this capability may be provided on a personal computer (not without precedent, e.g., Bertrand, 1980, or Lefkovitz, 1982).

We proceed by designing a model of the "technology transfer workstation", identifying commercial and/or public-domain offerings which can be used to satisfy the modular requirements of the architecture, and/or developing those modules which cannot be obtained in other ways.

2.1 An Architectural Model

We decompose this technology transfer process into four basic functions:

Defining
Searching
Sorting
and Writing.

"Defining" is the earliest phase in the technology transfer process. It is at this stage that it is realized that searching for a technological solution to a problem in a formal, explicit manner is appropriate and desirable. And, it is at this stage, that the problem is defined, that is, codified and written down in such a way that a professional database searcher and a professional engineer can address it. This defining stage is usually regarded as a "marketing" function in the organizations we are concerned with and is usually carried out by "marketing" personnel who are probably not the same people involved in the rest of the steps.

"Searching" refers to the identification of the databases which might be used to obtain relevant information, to the identification of keywords and key phrases which might be used to drive the search, and to the actual on-line search process.

"Sorting" refers to the task of culling and classifying the citations which are the result of the search into a structure which facilitates study and the eventual production of a written report.

"Writing" is the phase which results in the finished report product of this process of technology transfer.

Feedback certainly exists between these four steps. However, the better the interface and the cleaner the division between these phases, the better and cheaper the search is likely to be. Strong divisions between the steps in the presence of large volumes of requests for technology searches makes possible a division of labor and specialization which has been found to be beneficial to this process. In one case, at least, this division of labor is spread between marketing personnel, for defining the search, a professional bibliographic database searcher, for doing the actual online search, and professional engineers with technical specialties, for the sorting of the citations and the writing of the report.

The software components of our technology transfer workstation were selected and constructed to conform to this natural functional decomposition of this process.

2.2 User facility or Common Command Language

The searcher needs a user facility which gives him access to the online systems via telephone communications so that he can run searches, use the other features of the online system, and download citations. This need as so far described can be met by any of a large number of commercial and public-domain "terminal-emulation programs" which can run on the searcher's local computer.

In addition to this basic capability, it is desirable that the various online database systems be accessible through one common language for specifying searches, downloading citations, and manipulating the resulting files on his local computer. It is possible to obtain this common interface with a program running on the user's local computer. CONIT.(Marcus, 1982) is the only program which we have found which meets this need adequately, but CONIT does not seem to be moveable from its present, single (large computer) installation. IN-SEARCH (Menlo,

1984) is a commercial offering which works with only some databases and is not user-extendible.

Our present developmental activity in this project is confined mainly to this user interface area. Due to the interest expressed by several parties in the improvement of novice-user facilities for bibliographic databases, we consider this "executable documentation" approach to be intriqueing.

Presently, we are exploring the opportunities afforded by commercial pop-up scratchoad and keyboard command key macro packages for delivery of script-type user assistance. In addition, we are examining the dimensions opened up by structure-oriented programming editors and the new outlining software for script presentation. Some of the newer terminal emulation communication software for personal computers is beginning to incorporate this type of capability. We expect this to exploration to result in the development of a package (to be called SEARCH-AID) which distills and simplifies the most useful of these devices.

2.3 Reviewing and Determining Relevancy of Abstracts

A large number of citations can result from a single search. It is sometimes a large task to classify these citations as to relevancy or sub-category, e.g., Marcus,(1978). The TIS GATEWAY, Hampel, et al, (1982), offers some capability in this area in the form of tools for manipulating the abstracts and classifying them. In addition, the TIS GATEWAY offers some unique tools for the analysis of citations by time of publication.

We have developed our SORT-AID system, which supports online reviewing of down-loaded abstracts. Classification categories can be supplied interactively by the searcher. These tools allow relevancy determination to be carried out in a "semi-automatic" mode, with more or less reliance on the computer, as the searcher chooses, and as his particular search justifies.

2.4 Translation of Citation/Abstract Formats

In preparation for presenting the results of the database search to the client, it is sometimes desirable to translate the citations from the various databases into one common format.

2.5 Organization and Preparation of Report

In the area of writing aids, the offerings on the personal computers have out-paced anything before seen. Outlining, organizing, and word-processing tools abound. The searchers should certainly have this capability.

There is the necessity of converting the files of citations as down-loaded from the online system into a format which can be included into the word-processor. Some word-processors have facilities for doing this; others do not, and it must be programmed.

Ultimately, work in this area may lead to some type of automatic presentation of search results. It is intended that commercial word-processors be used with the technology transfer workstation.

3. DEVELOPMENT OF THE WORKSTATION USER FACILITY

3.1 Database User Facilities

A "database user facility" is a classical fourth-generation component. As database menagement systems have evolved, they have shown greater and greater functionality to the user. Some of that functionality, i.e., terminal-specific functions such as graphics and lightpens, could be handled more effectively if they were isolated logically from the database management system itself. This approach resulted in specific software components, which can be termed "database user facilities".

Our user facility work has centered on placing appropriate extra-database system functionality in a personal computer on the searcher's desk. This functionality includes aspects of a command language along with the usual communication functions such as down-loading citation files.

A search for instances of the "user facility" found the systems: CONIT (Marcus, 1982), DIVERSE (Hambeigner, 1984), IN-SEARCH (Menlo, 1984), NAM (Treu, 1982), RIG (Lantz, 1982), and ZOG (Robertson, 1980). The currency of the citations and their sources suggests that our efforts are state-of-the-art.

3.2 Problem Statement

There are many bibliographic citation databases available to computer terminal equipment over telephone lines. These databases are useful for technological research and the development of bibliographies. Usually, for a researcher to use these databases, he must seek the assistance of an "expert intermediary" who specializes in the details of online searching and knows the command languages of the different systems.

We wish to design a user-interface for these information retrieval systems which achieves a common command language for several databases. This user-interface could reside in a personal or timeshared computer which is to act as a "gateway" to the bibliographic database systems. This user-interface is to mediate the dialog between the user and the online systems. This system could function in a menu-based manner. Integral help facilities are desirable. An important feature is that user extension be allowed. The system should have modes of usage for expert and non-expert users (Carroll, 1984).

There are many subtelties of using the various databases that are not apparent in the manuals. Developing truly equivalent commands which remain equivalent across several databases may not be possible. However, before searching can be done by non-experts, this must be accomplished. One way this work can be done is to provide the expert searchers with a vecihcle to codify such efforts and make them available to non-experts.

3.3 PCCL Program

Our first experiments were with a program which we developed called PCCL (Prototype Common Command Language). PCCL allows the building of custom menus and the providing of down-loadable command sets which can be called up by those menus. The menus and the associated command sets are accessible to the user.

Using PCCL, example menus were set up for common searching functions. The menus appeared the same to the user, but had different command sets underneath depending on the database system which was the target. Thus, template commands could be prepared by an expert searcher, coded into the PCCL format, and given over to a less-expert user for execution.

The PCCL program demonstrated the value and feasibility of removing the syntax barrier. The program also established the idea of the preparation of these "canned query template" by an expert, who is familiar with the nuances of the databases of interest and with the purposes of the specific less-expert user.

3.4 DBUF Program

The problem with the PCCL framework was that it was difficult to build enough intelligence into the command sets to accomplish more than the most rudimentary operations. Also, there was little tutorial value to using the program. The DBUF (Database User Facility) program was developed to alleviate the

PCCL shortcomings and to extend the PCCL strengths.

DBUF is a combination between a screen-oriented, structured-text editor and a communications program. Text may be entered and structured in outline form. It may be viewed at any level in the outline. Specific lines in the text may be designated for communication down the phone line. Before communication, those lines may be changed temporarily.

DBUF allows an expert searcher to develop "scripts" of particular examples of the searching process. These scripts may be indexed and outlined using structure editor features. These scripts can contain descriptions, rationales, specific command lines, aphorisms, and other searching wisdom.

The user can search his scripts for the one that applies to his problem and then follow it, modifying commands as required, and sending the appropriate commands down the line to the online database system.

Scripts can be developed and traded. An expert searcher who is not familiar with a new database might request a script from a colleague who is familiar with that database. Scripts might be catalogued by NASA and stored on a database for retrieval like citations. Scripts can be classified for novices, journeymen searchers, and so forth.

In this manner, the DBUF can be the basis for a common script command language. Parallel scripts can be developed for all of the database systems. Searchers can follow the scripts and beware the tips embedded in the scripts for the particular databases. In following the scripts the user gains expertise.

More expert users can develop their own shorthand note scripts. Expert users can publish and trade scripts.

The DBUF methods for "scripting" can be extended to provide "executable documentation." The user manuals for the online database systems are usually replete with example commands. If the complete texts of the manuals were rendered machine-readable and a suitable structure applied (possibly automatically), they could serve as scripts in DBUF. This offers exciting potential for opening up the database systems to more users.

There is some recent activity in annotating user documentation for machine-aided reviewing in the literature, e.g., Bramwell, (1983), Rouse, (1980), and Witten, (1985).

4. DEVELOPMENT OF WORKSTATION ABSTRACT REVIEWING COMPONENT

The SORT-AID system is a set of programs designed to be used by an engineer/researcher in the preparation of "industrial applications studies." The SORT-AID system is designed to be useful in that interim stage in report preparation after the database querying and searching have been completed and before the actual writing process begins. Accordingly, SORT-AID provides facilities for classifying, manipulating, and organizing the abstract/citation files resulting from online database searching.

4.1 Description of Operation of Program: NABST

Input is the individual files resulting from multiple database searches. These files are combined by the NABST program. After the combined file is created by NABST, it can be accessed immediately by the user with the REVIEW program. Optionally, an automatic or semi-automatic relevance ordering can be created for the combined citation file with the RANK program. After the abstract/citations in a combined file have been classified by the user into report-specific categories, they can be printed according to those categories with PRINT.

4.2 Description of Operation of Program: REVIEW

REVIEW reads the combined abstract/citation file and allows the engineer/searcher to review the contents in its created order or according to a "relevance" order as determined by the RANK program. Optionally the abstracts may be reviewed in the order they were received. The abstracts are presented in the format they were received in (at the request of the users of the system).

REVIEW asks the user which of several allowed for terminal types he is using. If the user has none of the types on the list, he should answer that he has teletype capability, which is one of the choices. The only use for this terminal information in REVIEW is to fill the screens from the top down rather than having them scroll from the bottom. This allows reading to begin immediately.

REVIEW is command-driven. An earlier version of SORT-AID was menu-driven. Users wanted it to be command-driven. Also, the addition of more functions in this version make a menu clumsy. The facilities offered are:

* proceed to next screen, which will be more of a multi-screen entry, or to the next entry.

* set abstract category, which allows the operator to assign a report-specific category or categories to be carried in the file with the citation. This category can be any character string without spaces or commas. Multiple categories are entered with commas in between and no spaces.

* go directly to the next abstract.

* go to the beginning of the current abstract.

* go directly to an abstract in the file addressed by its relative position in the file.

* back up one abstract in the file.

* search on the occurence of an entered string. This is useful for finding entries containing a particular word, phrase, or author name, for example.

* set search category; this allows the user to only see entries which have been previously set to a certain category or categories; "*" shows all categories, including un-categorized entries.

* delete an entry from the file.

* enter notes into the file as a discrete entry; this entry can be categorized and processed like the others.

* re-order an entry; this may be used to place a particularly interesting entry at the beginning of the file; or an uninteresting one at the end; this is done for the physical and the "relavence" orders.

* load an internal memory register with the relative position of the current abstract.

* return to the abstract pointed to by the internal memory register.

* stop processing.

REVIEW is designed to allow the searcher to review and categorize abstracs until he has enough for his report, at which time he can stop processing. It also allows categorization to proceed iteratively via stepwise refinement; that is, the searcher can go through the file once applying gross categories, and then begin at the beginning looking at only a single gross category, applying finer categories.

4.3 Description of Operation of Program: RANK

RANK may be executed after the combined abstract file is completely created by NABST. RANK creates a file which contains a record for each abstract in the combined file. Each record contains the relative position of its associated abstract. These records are ordered in decreasing order by a "relevance" score.

To create the "relevance" score, RANK chooses two sets of thirty words each and displays them to the user. The user can delete words from this set. The words are selected so as to represent the set of abstracts in the combined file. The user deletes those words which do not apply to the intended subject of his search. Using this information, RANK can calculate a "relevance" score.

4.4 Description of Operation of Program: PRINT

PRINT is invoked from within REVIEW and is used to include abstracts of particular categories in a report. The format of the abstracts may be changed, i.e., Chen (1985), and the page length and width may be specified. The abstracts are included in the "relevance" order or in physical order. PRINT is invoked by a command in the REVIEW program. Selection of abstracts for inclusion in the PRINT report follows the conventions set up in REVIEW.

5. ABSTRACT RELEVANCE DETERMINATION

Abstract relevance determination is a unique requirement of a technology transfer workstation. Our approach is to develop several tools which the searcher can choose according to his/her tastes and needs. One tool is the REVIEW program in SORT-AID, which allows the examination of each citation and a manual ranking/classification to be done.

Integrated with REVIEW in the SORT-AID system is the RANK program. RANK can order the abstracts according to its own measure of relevance. The searcher can then REVIEW these abstracts in the order, most relevant first, supplied by RANK.

Another tool, which exists now only as a PC-based prototype, is "hypertext"-derived. "Hypertext" refers to a multi-threaded indexing scheme for text. Our tool builds such a data structure for the down-loaded abstracts and allows the searcher to peruse them by navigating with terms.

5.1 Theory of Relevance Determination: RANK

The RANK program uses a method for semi-automatic relevance determination based on the lexical association methods of automatic indexing, i.e., Salton, (1975). This automatic indexing theory is applied to the abstract/citations which are already the result of a search, rather than to a collection as a whole, which is the context it was developed for. This "post-search" collection is comprised of abstract/citations of a homopeneous nature, as it is the product of a single qauery or related set of queries. this homogeneous nature of the collection accentuates the characteristics of the automatic indexing methods. The RANK program exploits this accentuation.

The method of relevance ranking employed may be considered to be similar to the keyword querying. However, the keywords are not selected from a controlled vocabulary of the database system, but from lists generated with automatic indexing algorithms applied internally to the "post-search" collection. The user is allowed to select his "query" keywords from these lists. The abstracts in the collection are ranked for relevance by the frequency of occurrence of these query keywords.

RANK presents two lists of automatically generated keywords to the searcher. The first list is based on collection frequency. This list contains the thirty most frequent words in the set of abstracts (which are not on the stop list). The user is allowed to mark which of these have high relevance to his problem. The second set of keywords is based on a signal-to-noise statistic. The user is allowed to mark which of these have high relevance to his problem. In determining a "relevance score" for each citation in the combined file, the score is increased by one for the presence of a collection frequency keyword marked relevant by the user, and is decreased by one for each signal-to-noise keyword NOT marked relevant by the user.

Indexing methods based on collection frequency have been found to produce high recall, but with low precision. Indexing methods based on signal-to-noise ratio produce precise output, but with lower recall. This method of semi-automatic relevance determination used in RANK combines the best features of these two methods.

Review allows the user to consider the entries for classification in descending order of relevance score. This implies that the user can terminate reviewing when, in his judgement, he has found enough material for his report and have some assurance that he has not overlooked many relevant entries. The savings is in how many abstract/citations this method allows him not to review. (To enforce such an economy, a cut-off score might be set and implemented in the programs, but this is not done now.)

5.2 Evaluation of Relevance Determination: RANK

With a test set of eight searches, the method used in RANK with no user determination of the relevance of the keywords has been found to be effective. This amounts to using RANK in a totally automatic mode. Used in this way, the number of entries considered to be relevant in the top ten selected by RANK is consistently 10 to 20% better than what would result from a random selection; the number of entries considered to be relevant in the top fifty selected by RANK is consistently 5 to 10% better than what would result from a random selection.

Currently, the potential of the method used in the semi-automatic mode (with user designation of relevant keywords) is being investigated. Theoretically, this mode can be very powerful, especially if used with the engineering-style objective of finding a small number of acceptable citations, and not establishing too high a regret on missing some in the process.

6. SUMMARY

Technology transfer activities are carried out by NASA via a network of Industrial Applications Centers and State Technology Assistance Centers. These organizations employ a number of modes of technology transfer which principally vary in detree of specificity. These modes are largely the same regardless of the technology under consideration. These processes are labor intensive and are now being automated using micro-computer-based techniques and software support systems. The software support systems encompass pre-processors for database access, gateway systems, post-processors for citation analysis, report organization and writing, text processing, communications, and program integration.

The present recommended configuration of the technology transfer workstation includes commercial communications program combined with a pop-up scratchpad package to support scripting, a

10 megabyte disk equipped IBM PC, a 1200 baud intelligent modem, SORT-AID, a software package developed by us for classifying downloaded abstracts, and a commercial wordprocessor. Currently, we are developing our own communications program which will support scripting, called SEARCH-AID, which will help replace the commercial communicating program and pop-up package.

After-the-fact, our methodology appears to be tailoring and applying generic PC software (such as communications programs and file managers) to this specific problem until the problem and the solution is understood. At that time, if justified, we develop a custom solution to the problem.

Evaluation of this technology transfer workstation will be carried out in the coming year.

7. REFERENCES AND BIBLIOGRAPHY

[1] Bertrand, D., and C. R. Bader, "Storage and Retrieval of Bibliographic References Using a Microprocessor System." Int. J. Bio-Medical Computing, vol. 11, 1980.

[2] Bramwell, Bob, "BROWSE: An On-line Manual and System Without an Acronym," Asterisk, December, 1983.

[3] Carroll, John M., and Caroline Carrithers. "Training Wheels in a User Interface", Comm. of the ACM, August, 1984.

[4] Chen, Violet, M.S. Project Report, "An Algorithm for Recognizing and Reformatting Substrings," University of Southern Mississippi, Department of Computer Science, 1985.

[5] Hampel, V.E., et al., "Down Loading and Post-Processing of Bibliographic Information with the TIS Intelligent Gateway Com-puter", On-Line '82 Conference, November 1-3, 1982.

[6] Dennis Heimbigner, "Towards an Integrated Environment for Accessing External Databases." PROCEEDINGS OF SECOND ACM-SIGOA CONFERENCE ON OFFICE INFORMATION SYSTEMS, June 25-27, 1984, Toronto, Canada, vol. 5 no. 1-2.

[7] Lantz, K.A., Klaus Gradischnig, Jerome A. Feldman, Richard Rashid, "Rochester's Intelligent Gateway," IEEE COMPUTER, October, 1982, p. 54-70.

[8] Lefkovitz, David, "PRIMATE: A Microcomputer System for Host Mediation and Private File Text Search." Proceedings of the ASIS Annual Meeting, vol. 19, 1982.

[9] Marcus, Richard S., Peter Kugel, and Alan R. Benenfeld. "Catalog Information and Text as Indicators of Relevance." Jasis, January, 1978.

[10] Marcus, Richard S., "User Assistance in Bibliographic Retrieval Networks through a Computer Intermediary." IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, vol. SMC-12, no. 2, March/April, 1982.

[11] Menlo Corporation, "IN-SEARCH"-An On-line Database Support Package", 1984.

[12] Robertson, D., D. McCracken and A. Newell. "The ZOG approach to man-machine communication." INTERNATION JOURNAL OF MAN-MACHINE STUDIES. (1981), 14, 461-488.

[13] Rouse, Sandra H., and William B. Rouse, "Computer-Based Manuals for Procedural Information," IEEE Trns. on Systems, Man, and Cybernetics, vol. SMC-10, August, 1980.

[14] Salton, G., "A Theory of Indexing", Society for Industrial and Applied Mathematics, Philadelphia, PA. 1975.

[15] Treu, Siegfried, "Uniformity in User-Computer Interaction Languages: A Compromise Solution," Int. J. Man-Machine Studies. vol. 16, 1982.

[16] Vital, Dennis, M.S. Thesis, "A Portable Database Searcher's Software Toolset", University of Southern Mississippi, Department of Computer Science, 1985.

[17] Witten, Ian H., and Bob Bramwell, "A System for Interactive Viewing of Structured Documents." Comm. of the ACM, March, 1985.